Machine learning
for better query planning

Oleg Ivanov

8th of April, 2016

Outline

1. Query planning
2. Machine learning for better query planning
3. Further research

PROFESSIONAL Postgres

SQL

```
SELECT *
FROM users
WHERE age > 25;
```

Users

id	name	age	city
0	Ivan	25	MSC
1	Petr	39	SPB
3	Sidor	14	MSC
4	Pavel	47	LON
5	Petr	15	MSC

Messages

sender_id	text	reciever_id
3	Hi!	5
5	Who r u?	3
3	I'm Sidor! :)	5
3	And you?	5

Result

id	name	age	city
1	Petr	39	SPB
4	Pavel	47	LON

SQL

```
SELECT *
FROM users, messages
WHERE age < 15 AND users.id = messages.sender_id;
```

Users

id	name	age	city
0	Ivan	25	MSC
1	Petr	39	SPB
3	Sidor	14	MSC
4	Pavel	47	LON
5	Petr	15	MSC

Messages

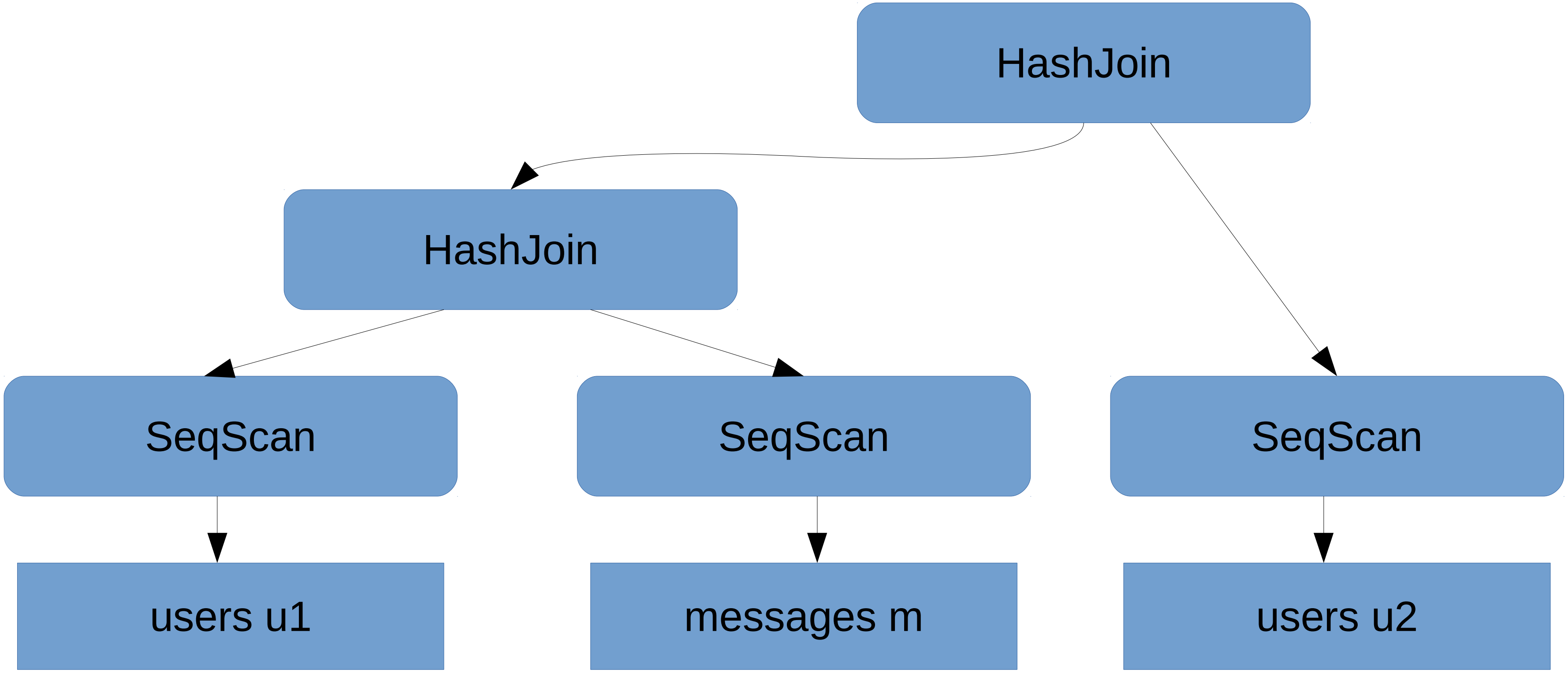
sender_id	text	reciever_id
3	Hi!	5
5	Who r u?	3
3	I'm Sidor! :)	5
3	And you?	5

Result

id	name	age	city	sender_id	text	reciever_id
3	Sidor	14	MSC	3	Hi!	5
3	Sidor	14	MSC	3	I'm Sidor! :)	5
3	Sidor	14	MSC	3	And you?	5

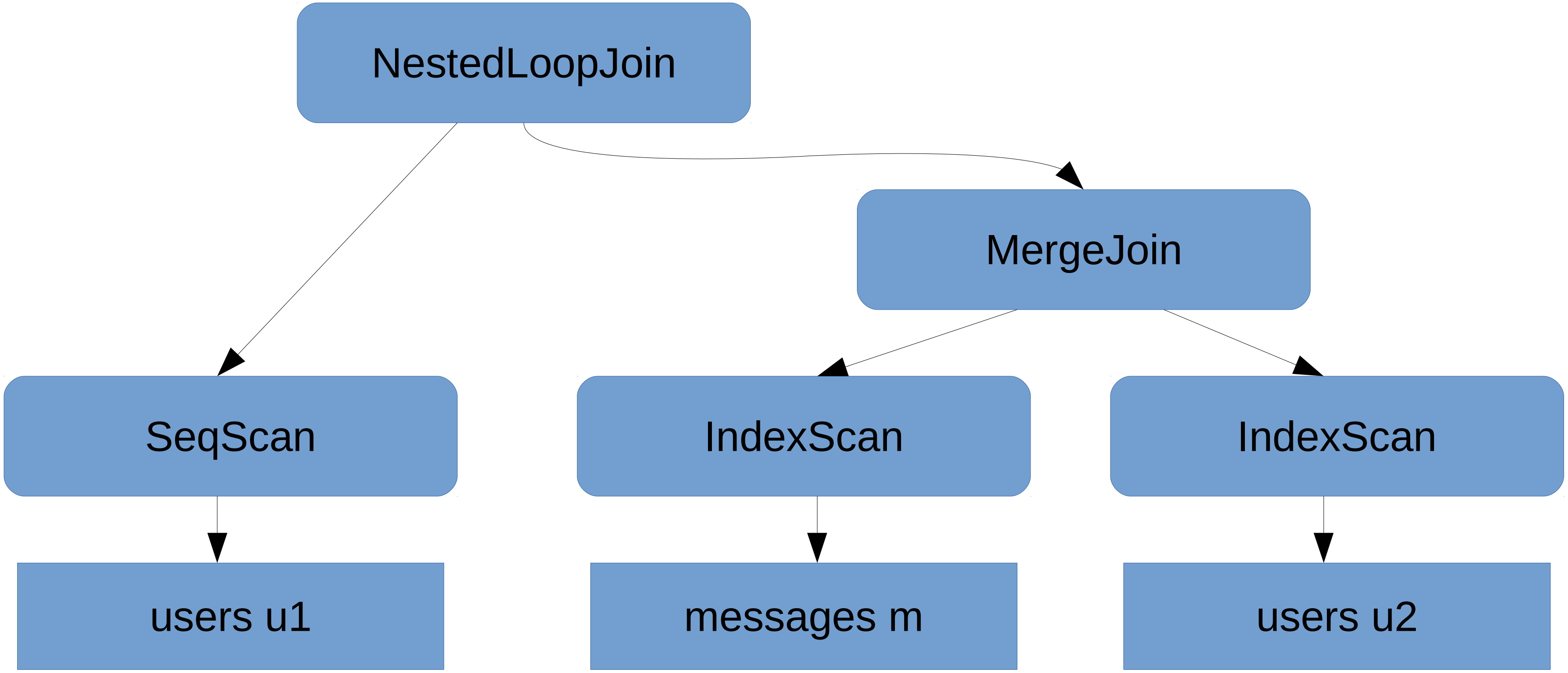
Query execution plan

SELECT *
FROM users **AS** u1, messages **AS** m, users **AS** u2
WHERE u1.id = m.sender_id **AND** m.receiver_id = u2.id;



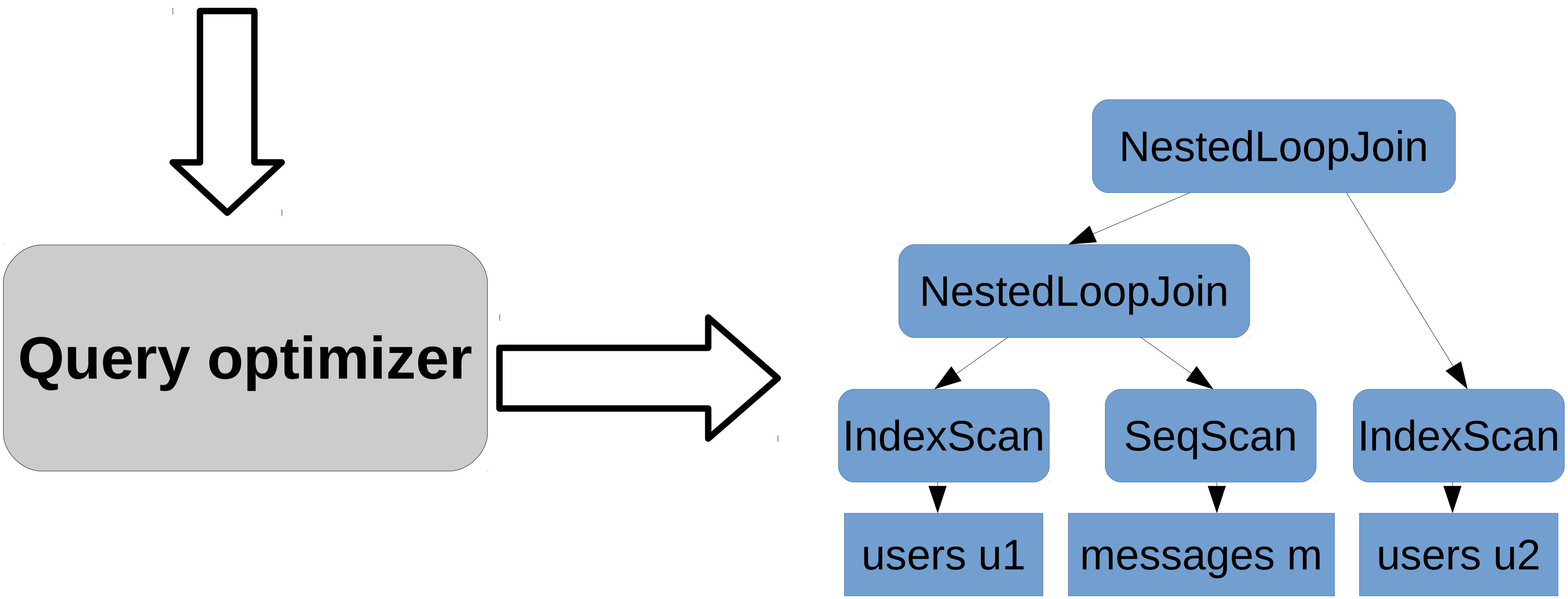
Query execution plan

```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



Motivation

```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



Relational DBMS

First relational DBMS:

IBM System R (1974)

Query optimizer:

- Rule-based
- Cost-based (System R)

Cost-based optimizer:

Cost estimation

Optimization over all possible plans

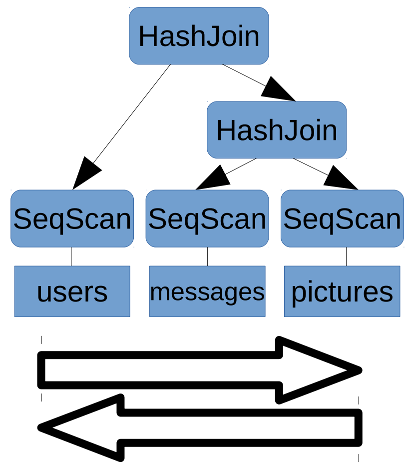
How to choose execution plan?

Optimization method

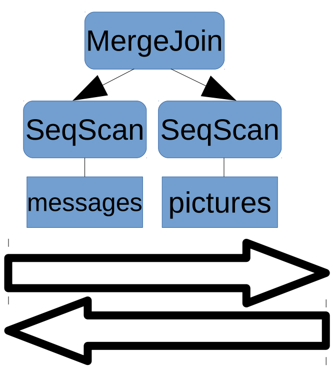
Dynamic programming

or

Genetic algorithm



439429 units



304528 units

Plan's cost estimation

Dynamic programming

Dynamic programming on subsets:

$$f(X) = \underset{x \subset X}{aggregate}\left(g\left(f(x), f(X \setminus x), x, X \setminus x\right)\right)$$

System R cost-based model:

$$cost(X) = \min_{x \subset X}\left(join\left(cost(x), cost(X \setminus x), x, X \setminus x\right)\right)$$

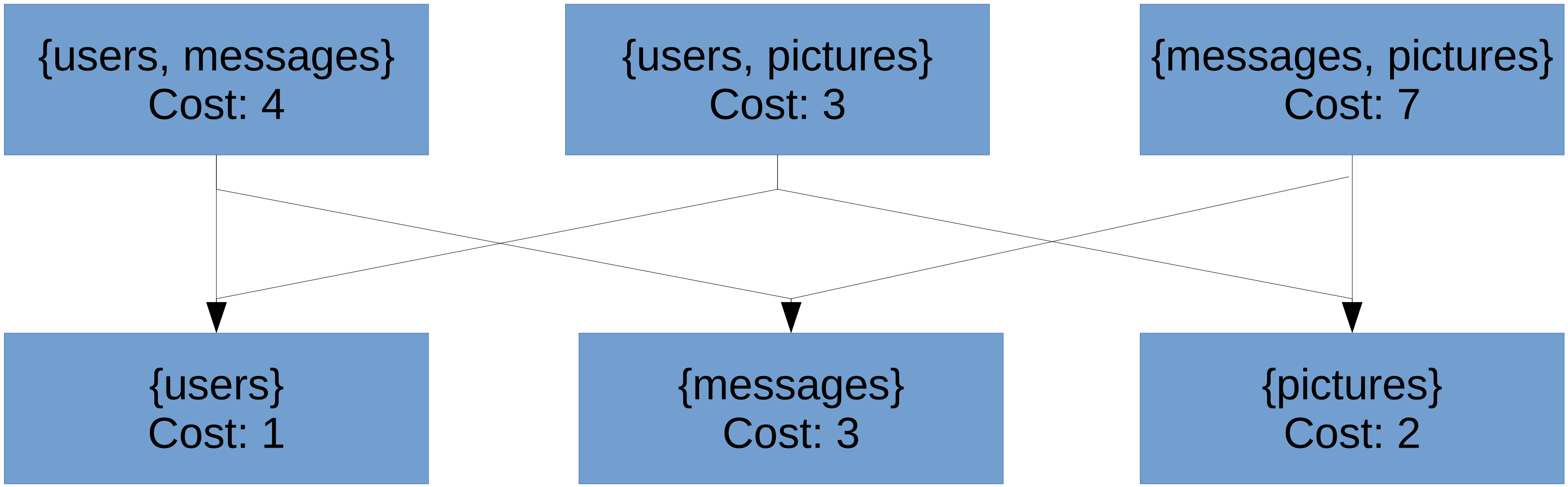
Dynamic programming

{users}
Cost: 1

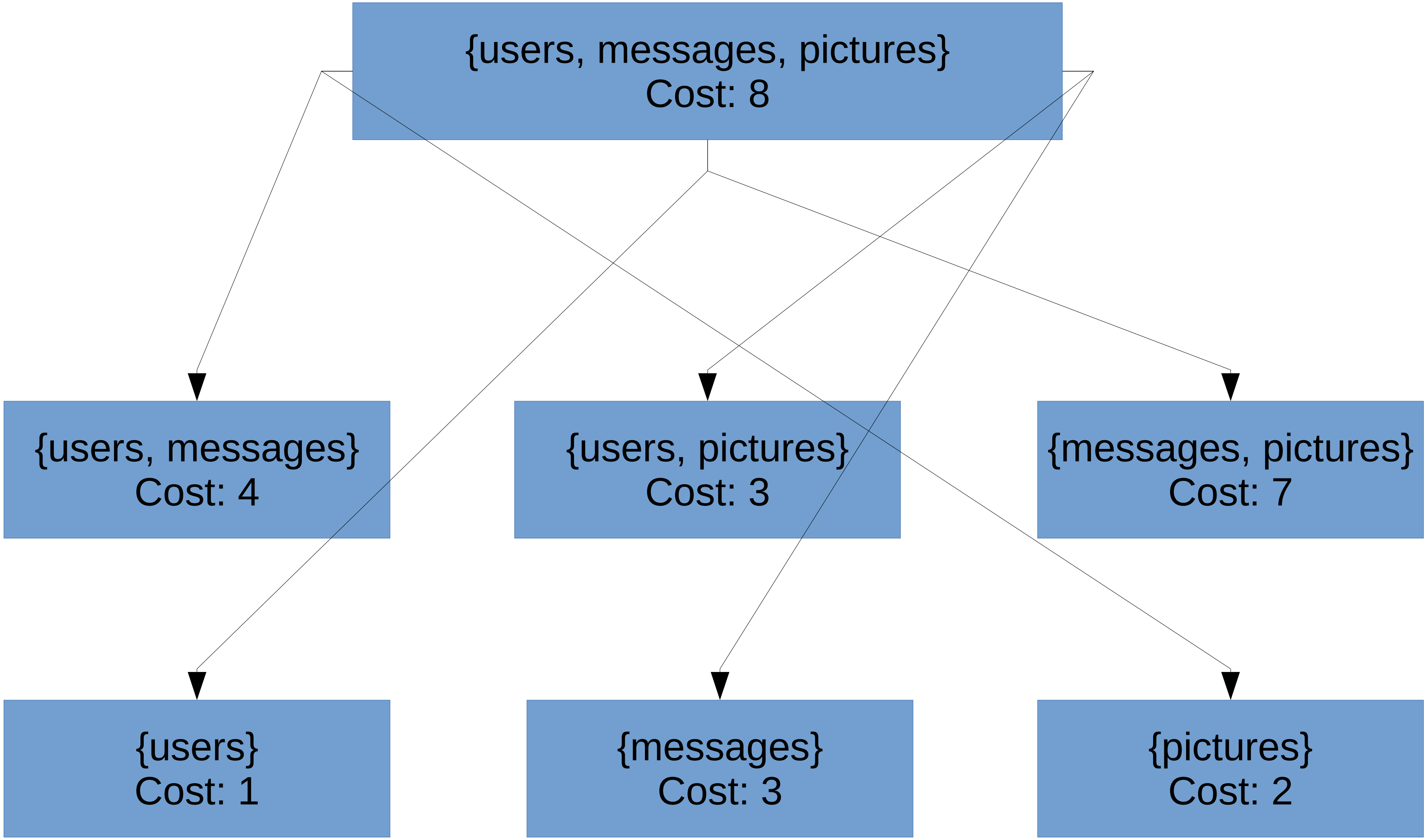
{messages}
Cost: 3

{pictures}
Cost: 2

Dynamic programming



Dynamic programming



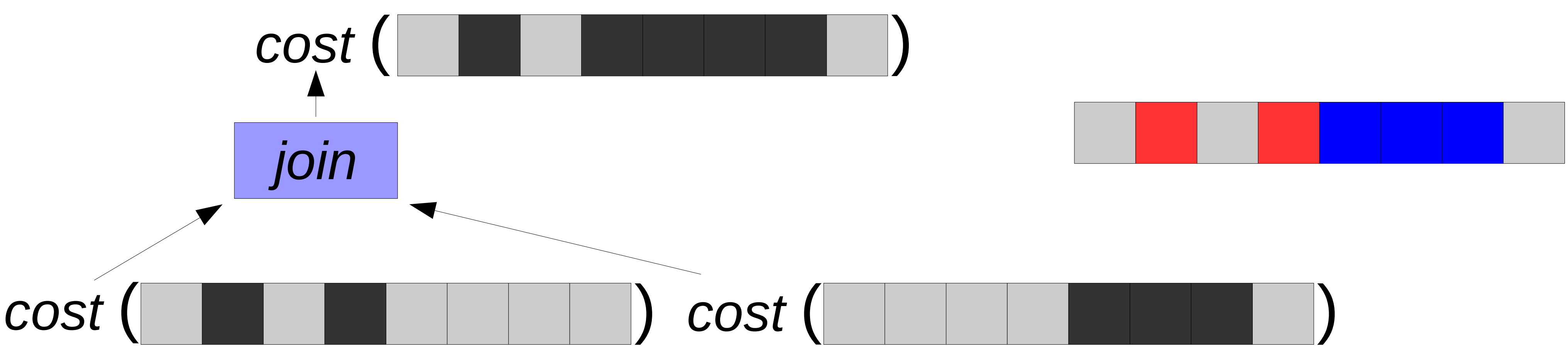
Dynamic programming

System R cost-based model:

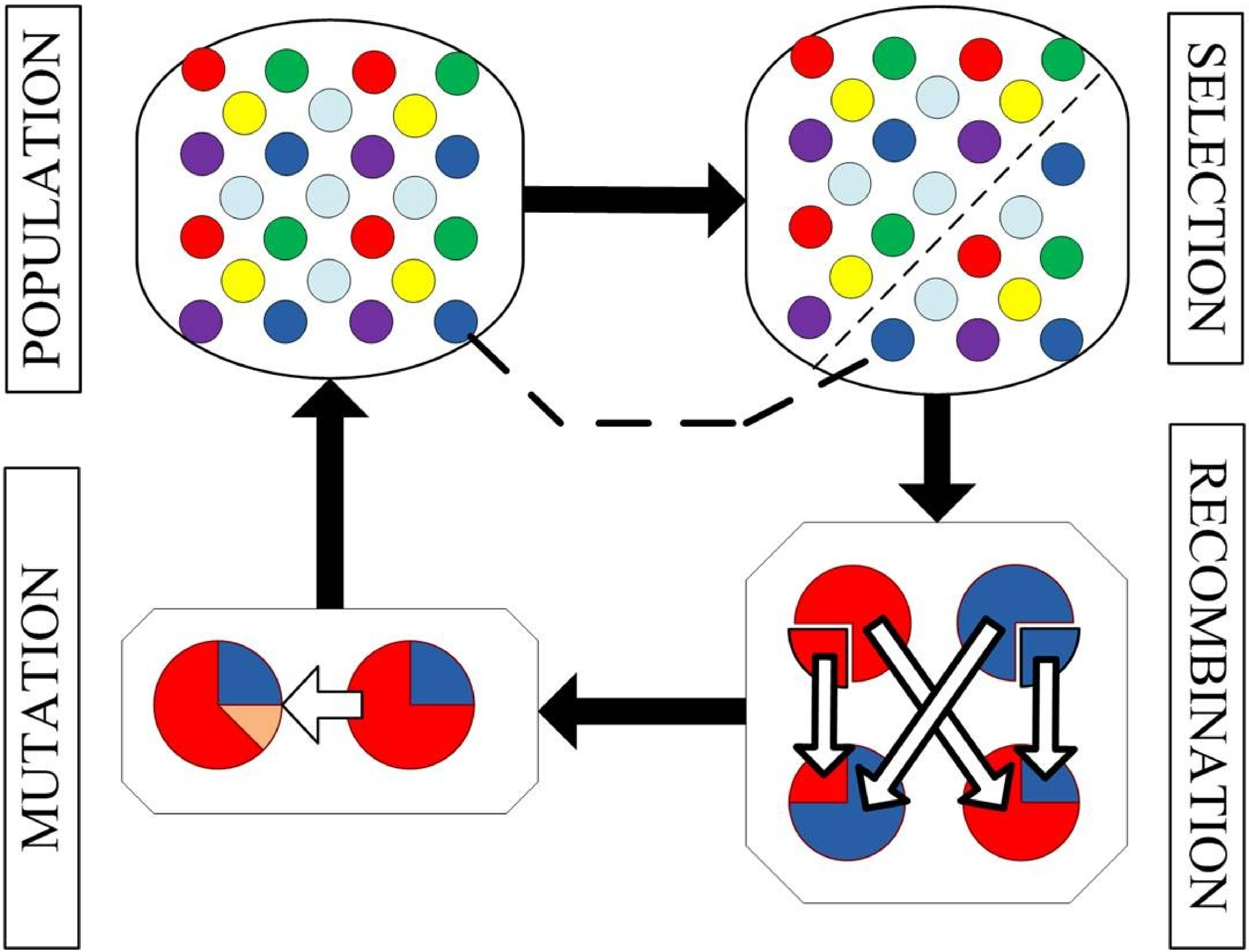
$$cost(X) = \min_{x \subset X}\left(join\left(cost(x), cost(X \setminus x), x, X \setminus x\right)\right)$$

Memory complexity: $O(2^N)$

Time complexity: $O(3^N)$



Genetic algorithm



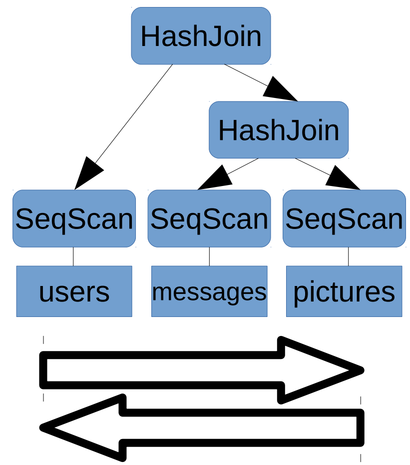
How to choose execution plan?

Optimization method

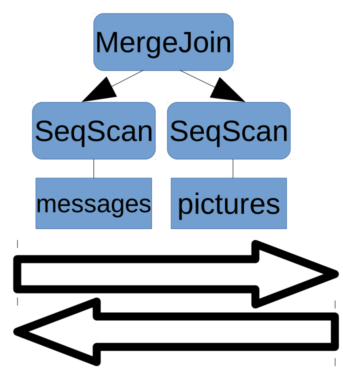
Dynamic programming

or

Genetic algorithm



439429 units

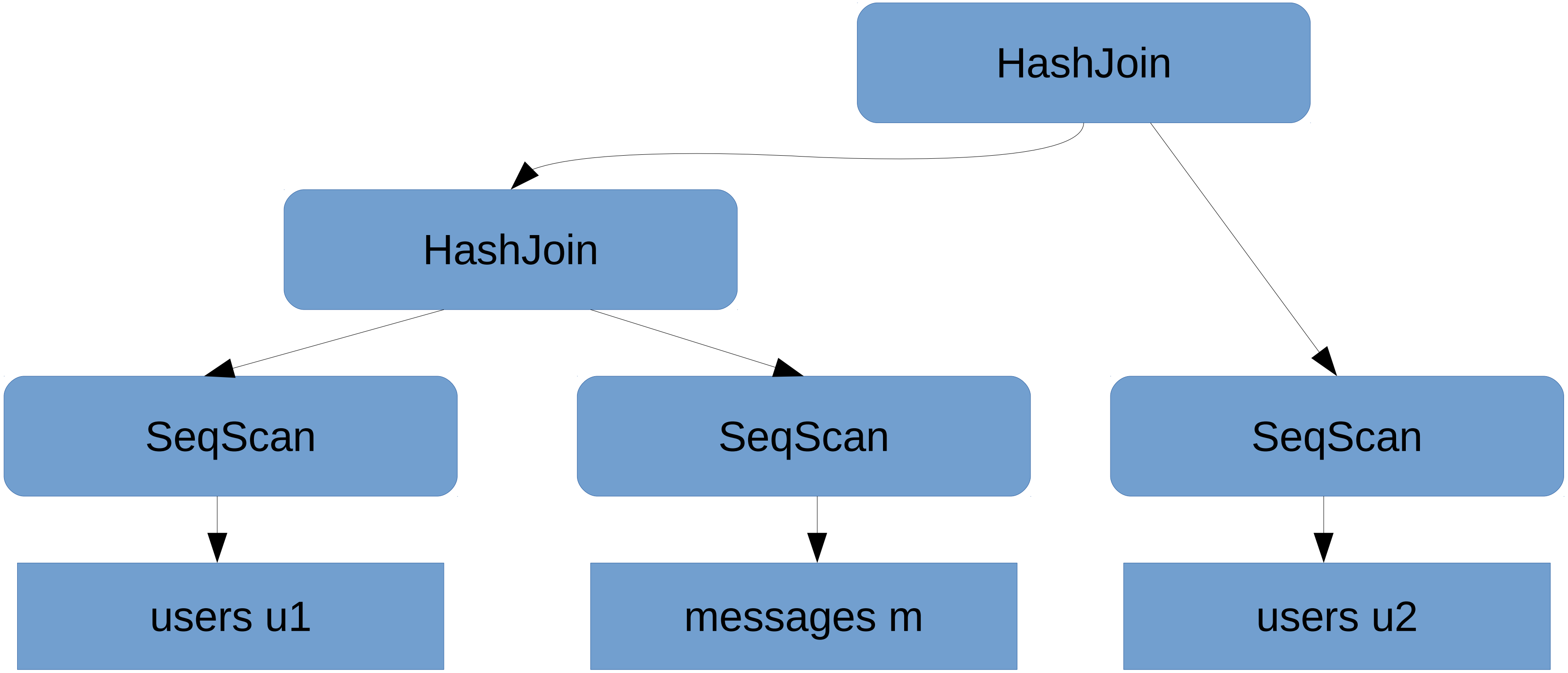


304528 units

Plan's cost estimation

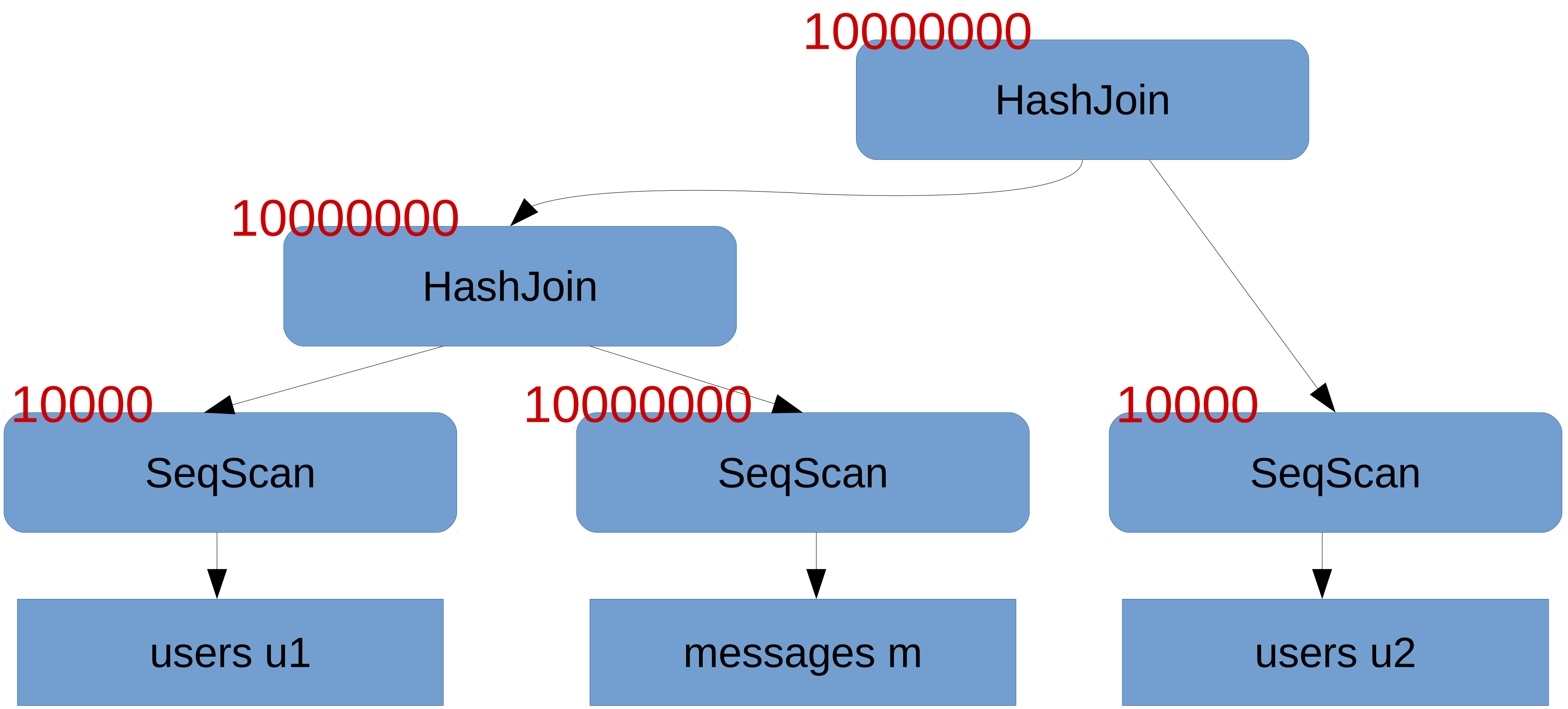
Cost estimation

```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



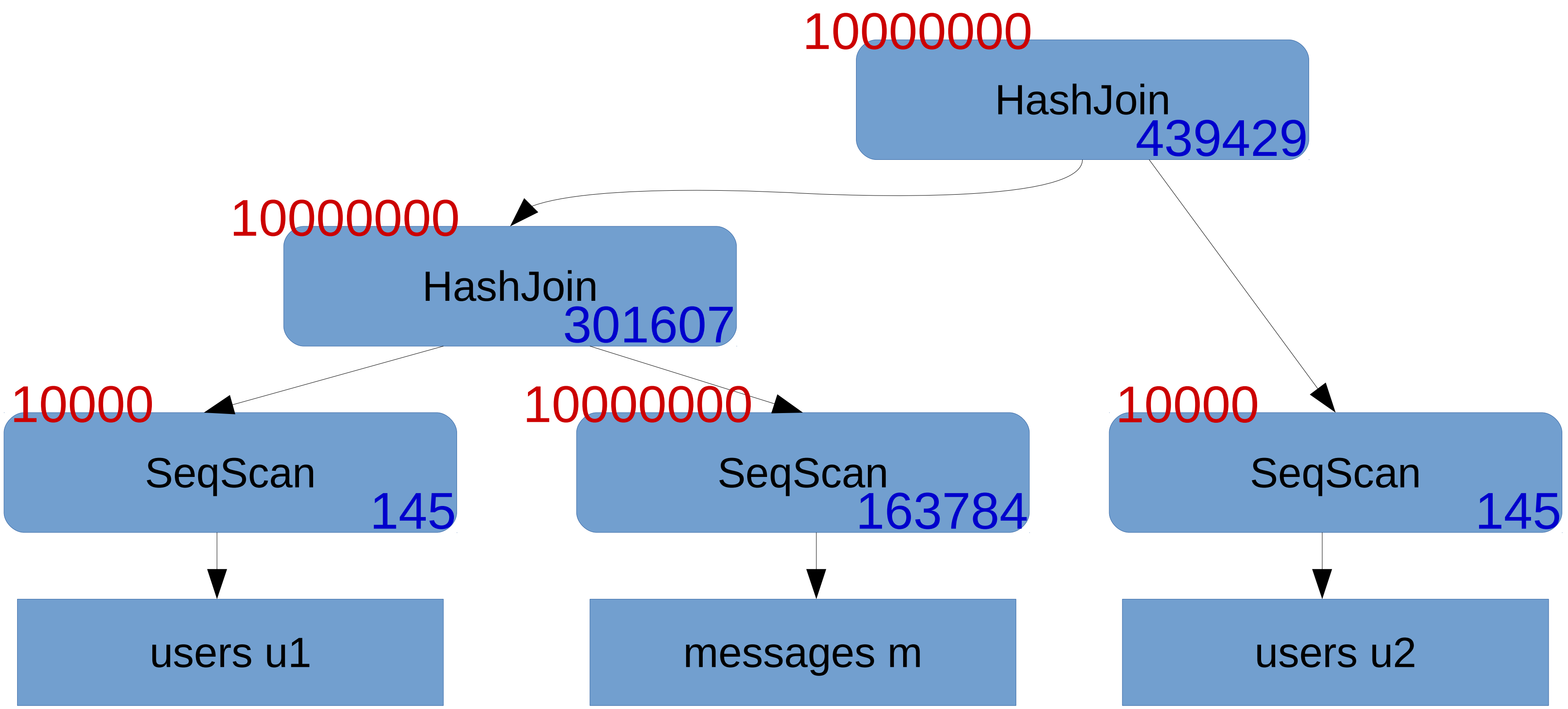
Cardinality estimation

```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



Cost estimation

```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



Cost estimation

$$C_O = n_s c_s + n_r c_r + n_t c_t + n_i c_i + n_o c_o$$

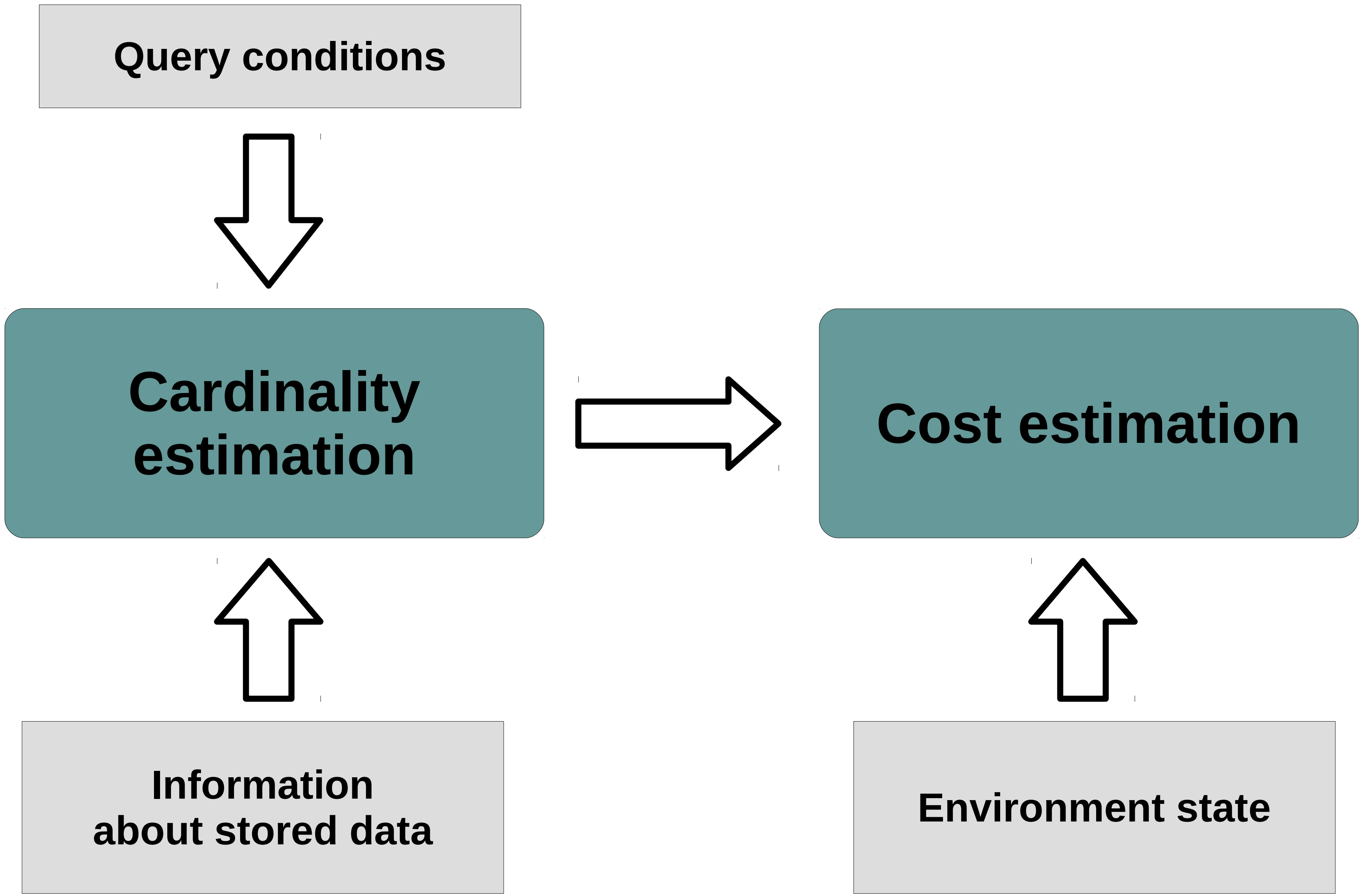
$$C = \sum_{O \in Tree} C_O$$

c_s	seq_page_cost	1.0
c_r	random_page_cost	4.0
c_t	cpu_Tuple_cost	0.01
c_i	cpu_Index_tuple_cost	0.005
c_o	cpu_Operator_cost	0.0025

In-memory sort:

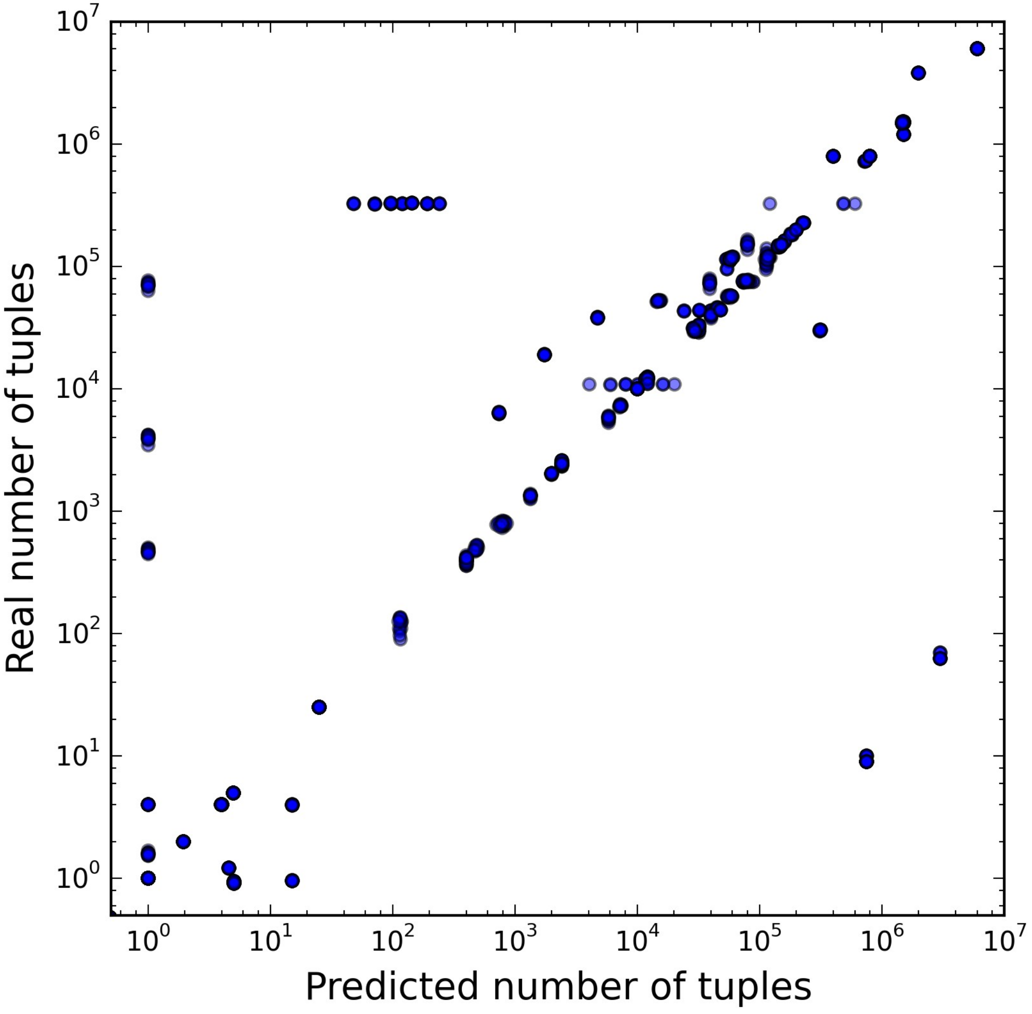
$$n_o = 2 \cdot 1.39 \cdot N \cdot \log_2 N + N$$

Cost estimation



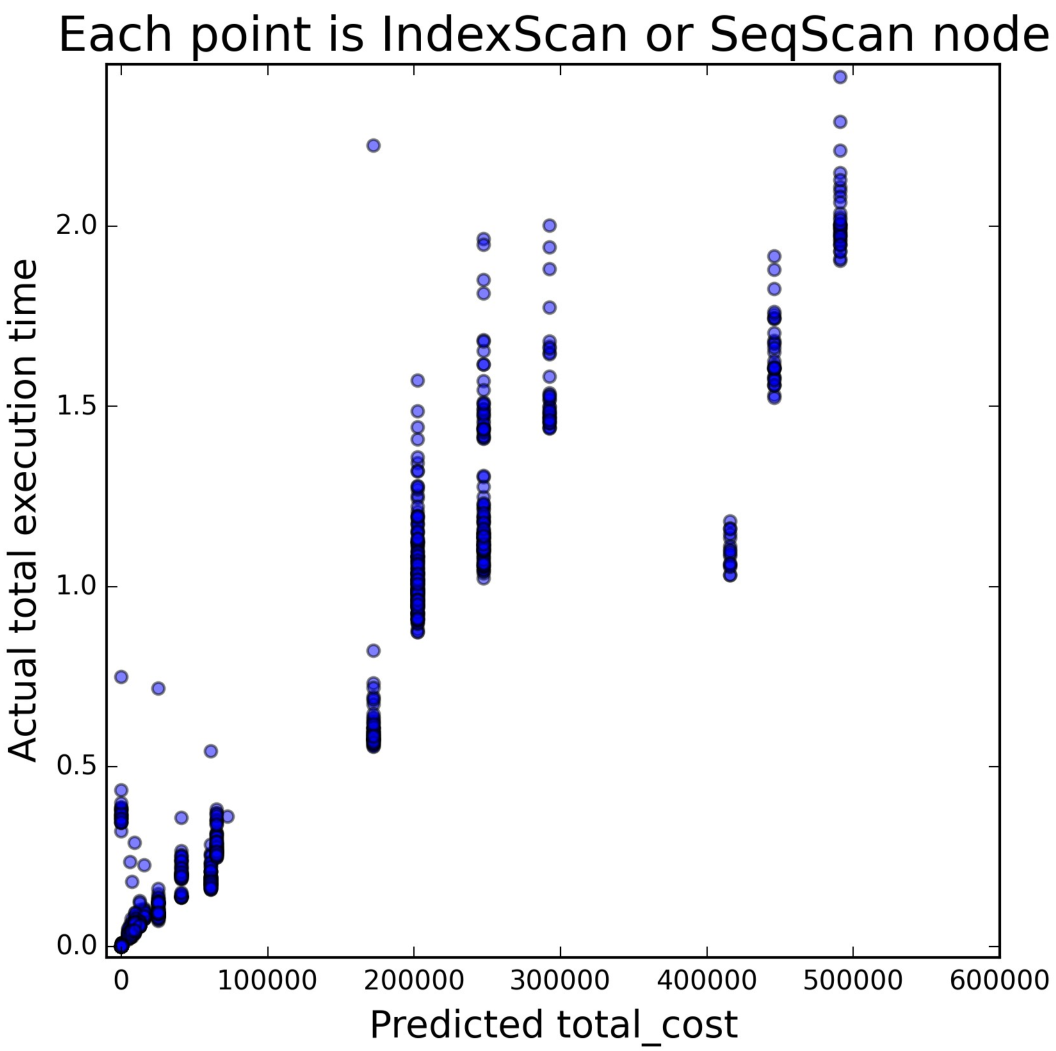
Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

Cardinality estimation



Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

Cost estimation



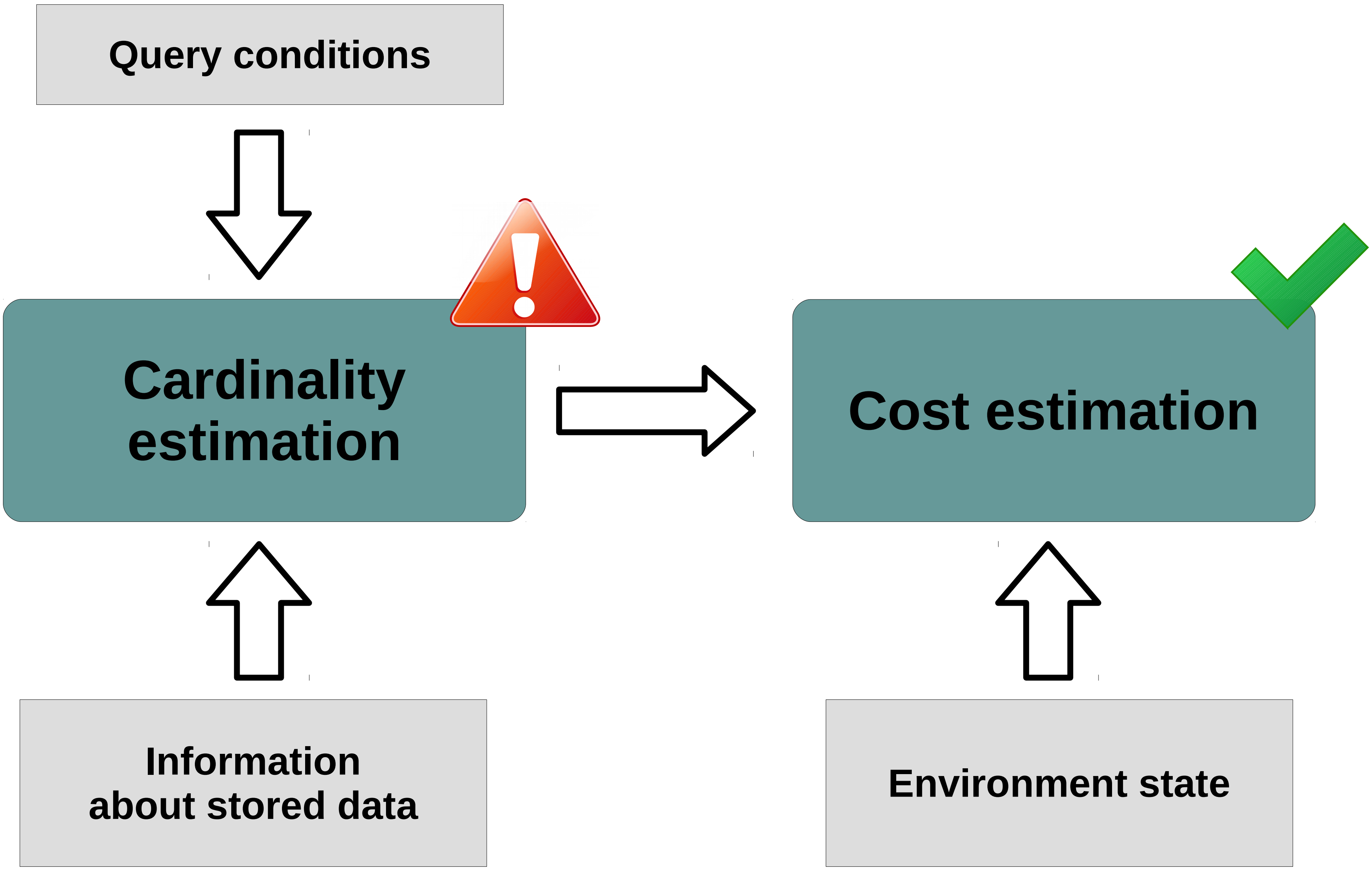
Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

Cost estimation

How good are query optimizers, really?

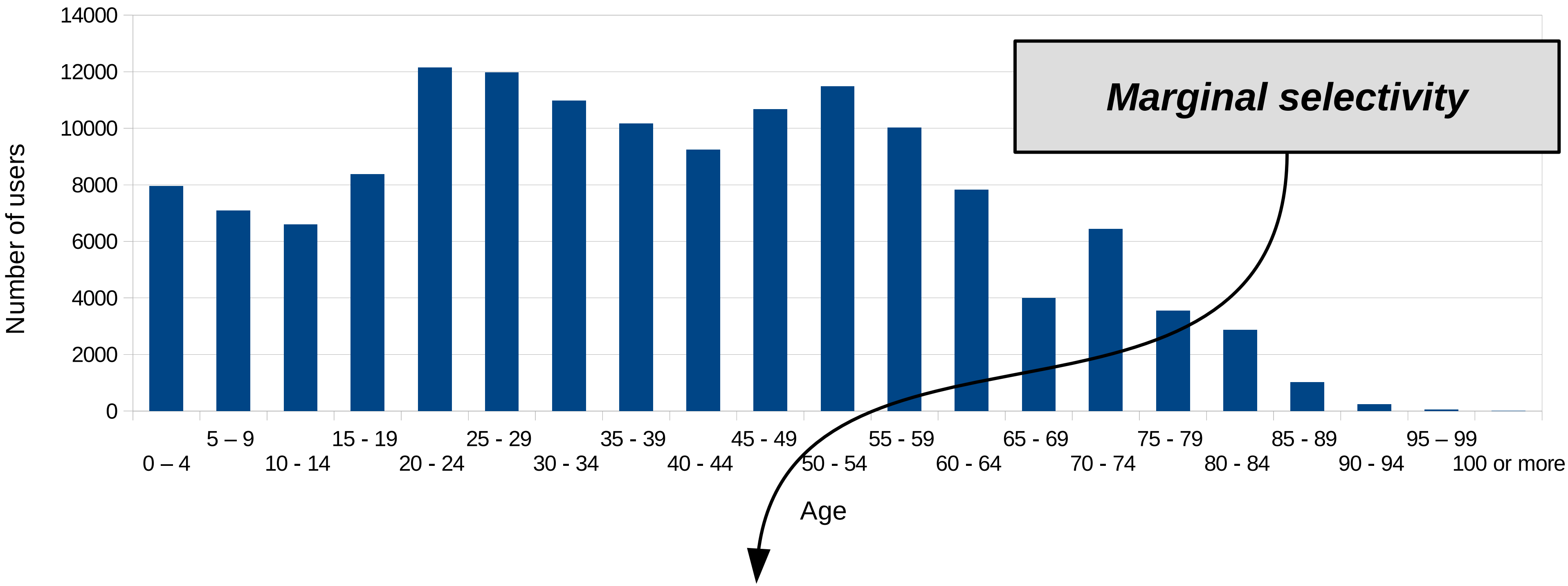
V. Leis, A. Gubichev, A. Mirchev et al.

Cost estimation



Cardinality estimation

SELECT * **FROM** users
WHERE age < 25;



$$Selectivity \simeq 0.3$$

$$Cardinality = Tuples \cdot Selectivity$$

Joint selectivity

```
SELECT * FROM users
WHERE age < 25 AND city = 'Moscow';
```

We have only marginal selectivities
The conditions are assumed to be independent

$$Selectivity_{age,city} = Selectivity_{age} \cdot Selectivity_{city}$$

Excluding $Selectivity_{25<age\ \mathrm{AND}\ age<57} = Selectivity_{25<age<57}$

Selectivity overestimation

```
SELECT * FROM users
WHERE position = 'cleaner' AND salary > 50000;
```

$$Selectivity_{cleaner} \simeq 0.2$$

$$Selectivity_{salary} \simeq 0.3$$

$$\cancel{Selectivity_{salary,cleaner} = Selectivity_{salary} \cdot Selectivity_{cleaner}}$$ **Wrong!**

$$Selectivity_{salary,cleaner} \simeq 0$$ **Correct**

Contradiction of conditions is not the common case

Selectivity underestimation

```
SELECT * FROM users
WHERE position = 'cleaner' AND salary < 50000;
```

$$Selectivity_{cleaner} \simeq 0.2$$

$$Selectivity_{salary} \simeq 0.3$$

$$\cancel{Selectivity_{salary,cleaner} = Selectivity_{salary} \cdot Selectivity_{cleaner}}$$ **Wrong!**

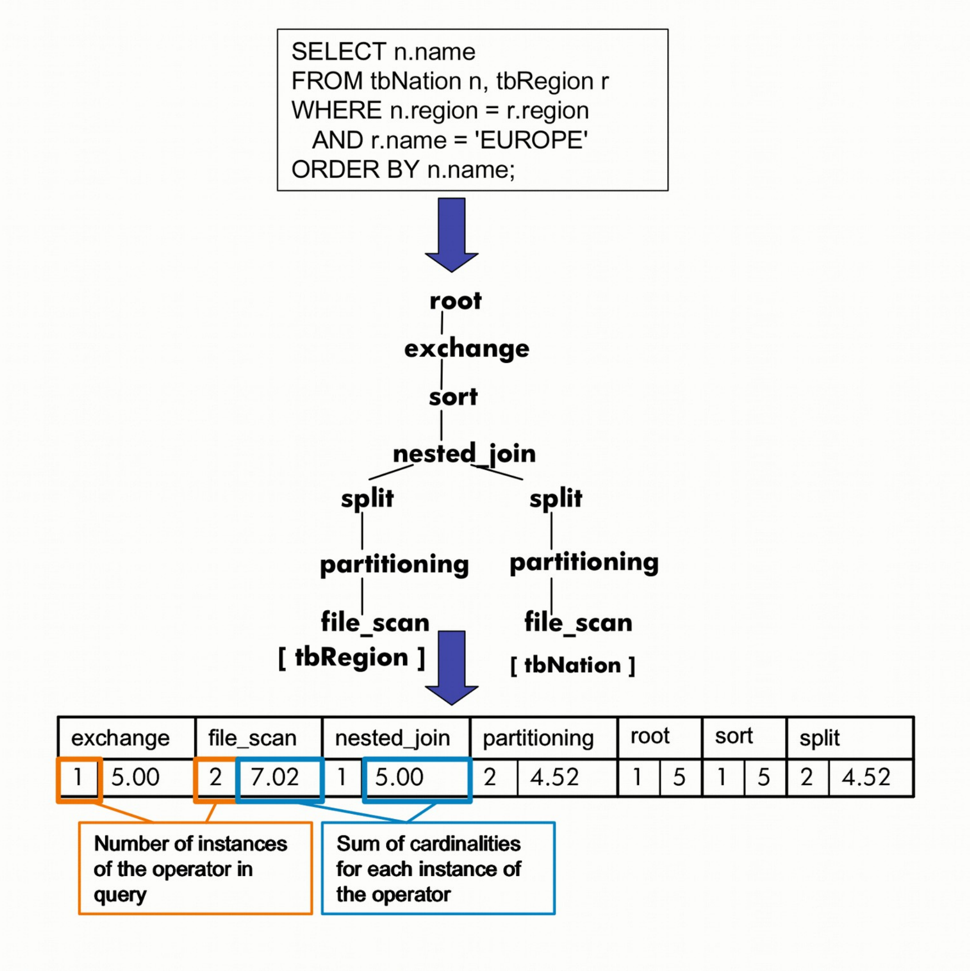
$$Selectivity_{salary,cleaner} \simeq Selectivity_{cleaner}$$ **Correct**

Common case is when a condition makes more precise previous ones

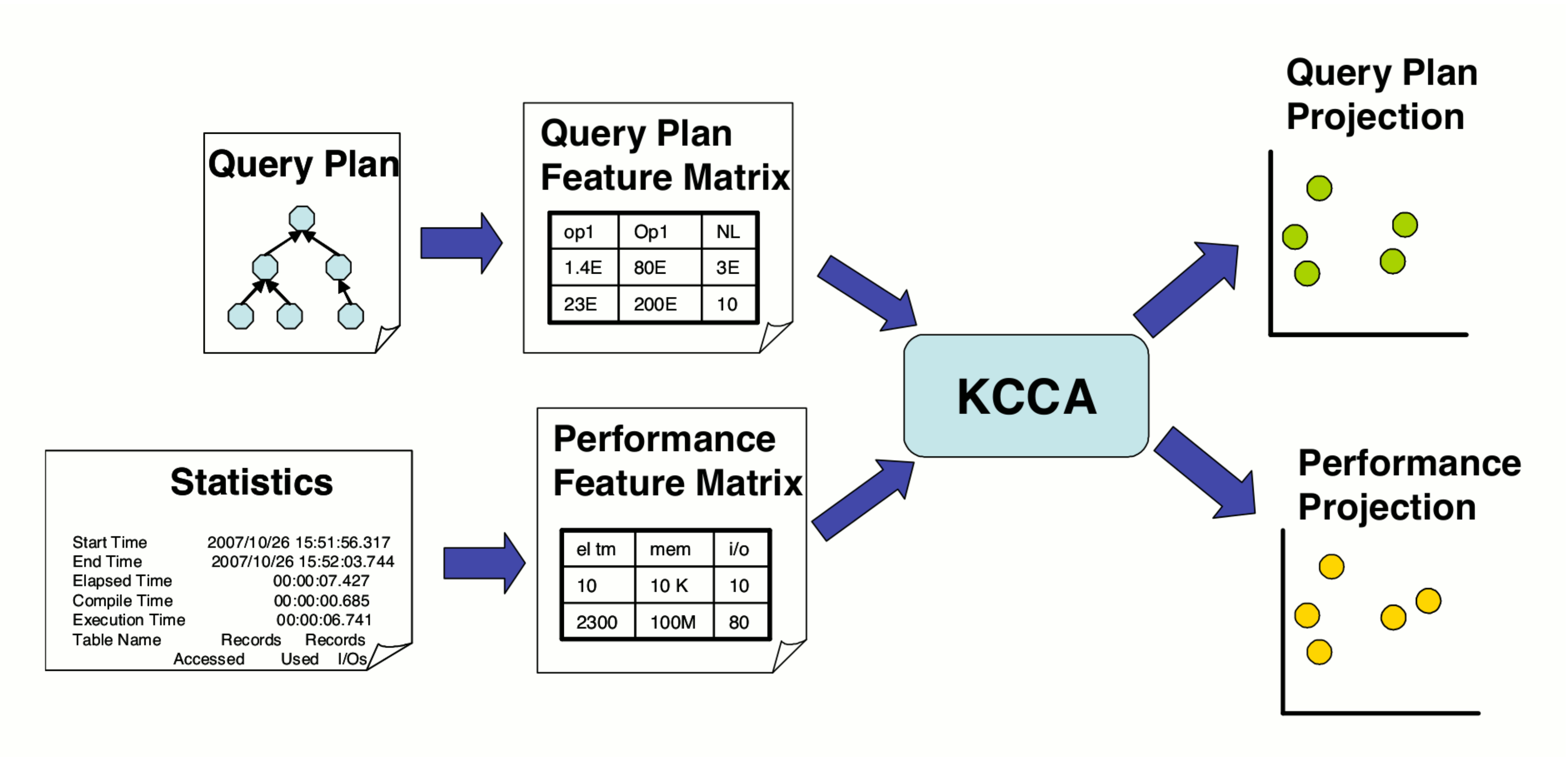
Related work

- Predicting multiple metrics for queries: Better decisions enabled by machine learning / A. Ganapathi, H. Kuno, U. Dayal et al.
- Learning-based query performance modeling and prediction / M. Akdere, U. Cetintemel, ¸M. Riondato et al.
- A machine learning approach to sparql query performance prediction / Hasan R., Gandon F.
- Robust estimation of resource consumption for sql queries using statistical techniques / J. Li, A. C. K¨onig, V. Narasayya, S. Chaudhuri
- Malik T., Burns R. C., Chawla N. V. A black-box approach to query cardinality estimation.

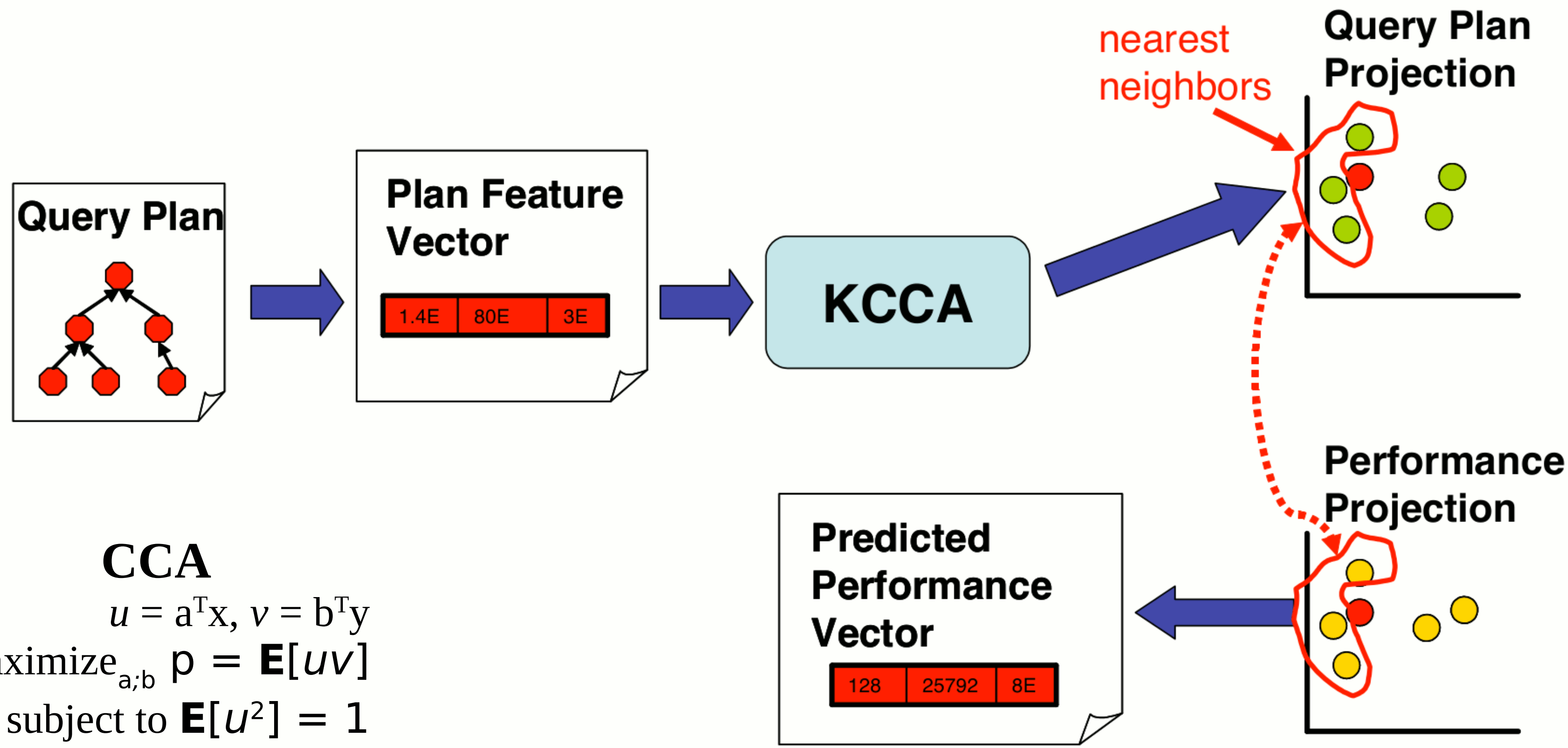
Related work



Related work



Related work



CCA

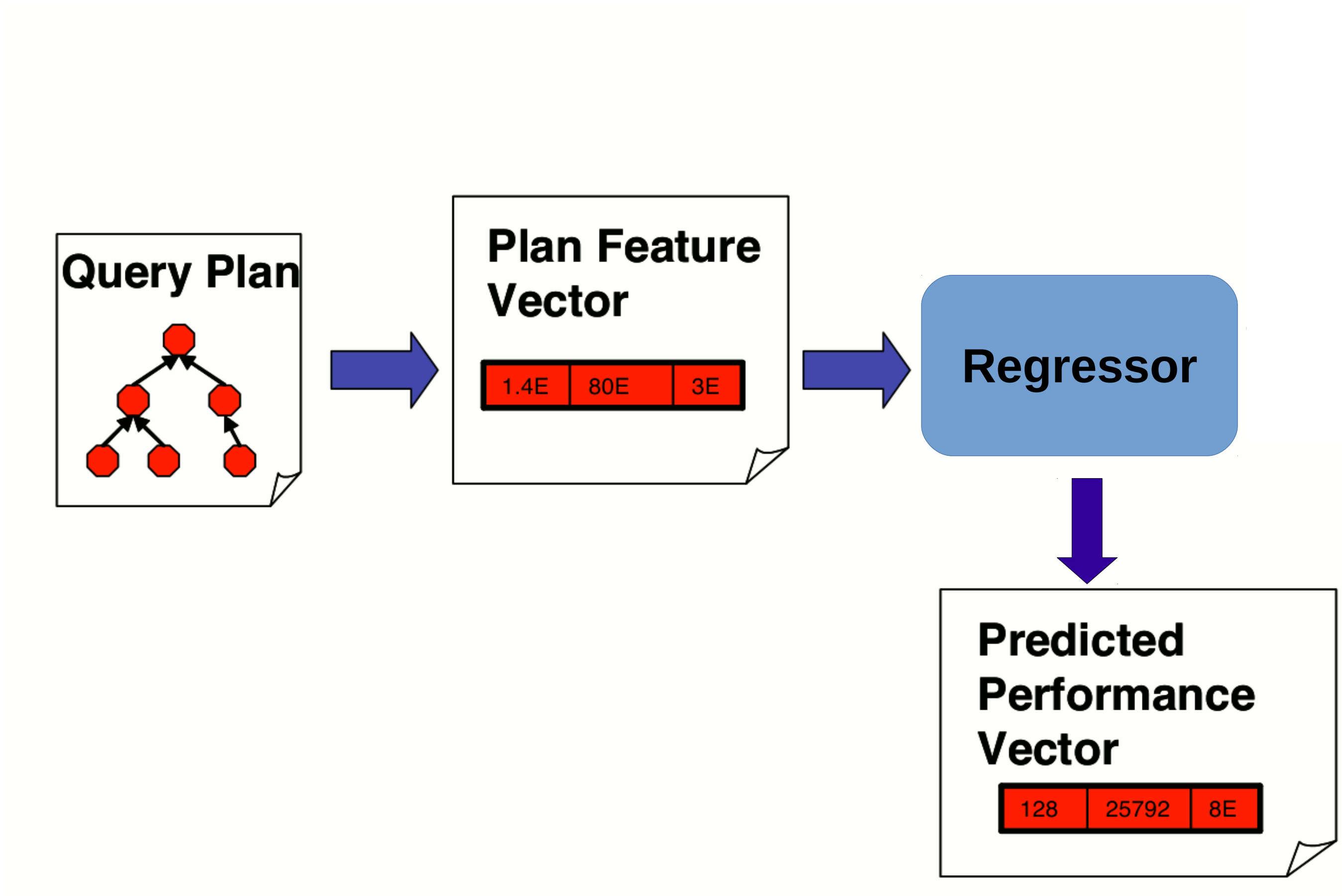
$$u = a^T x,\ v = b^T y$$

$$\text{maximize}_{a;b}\ \rho = \mathbf{E}[uv]$$

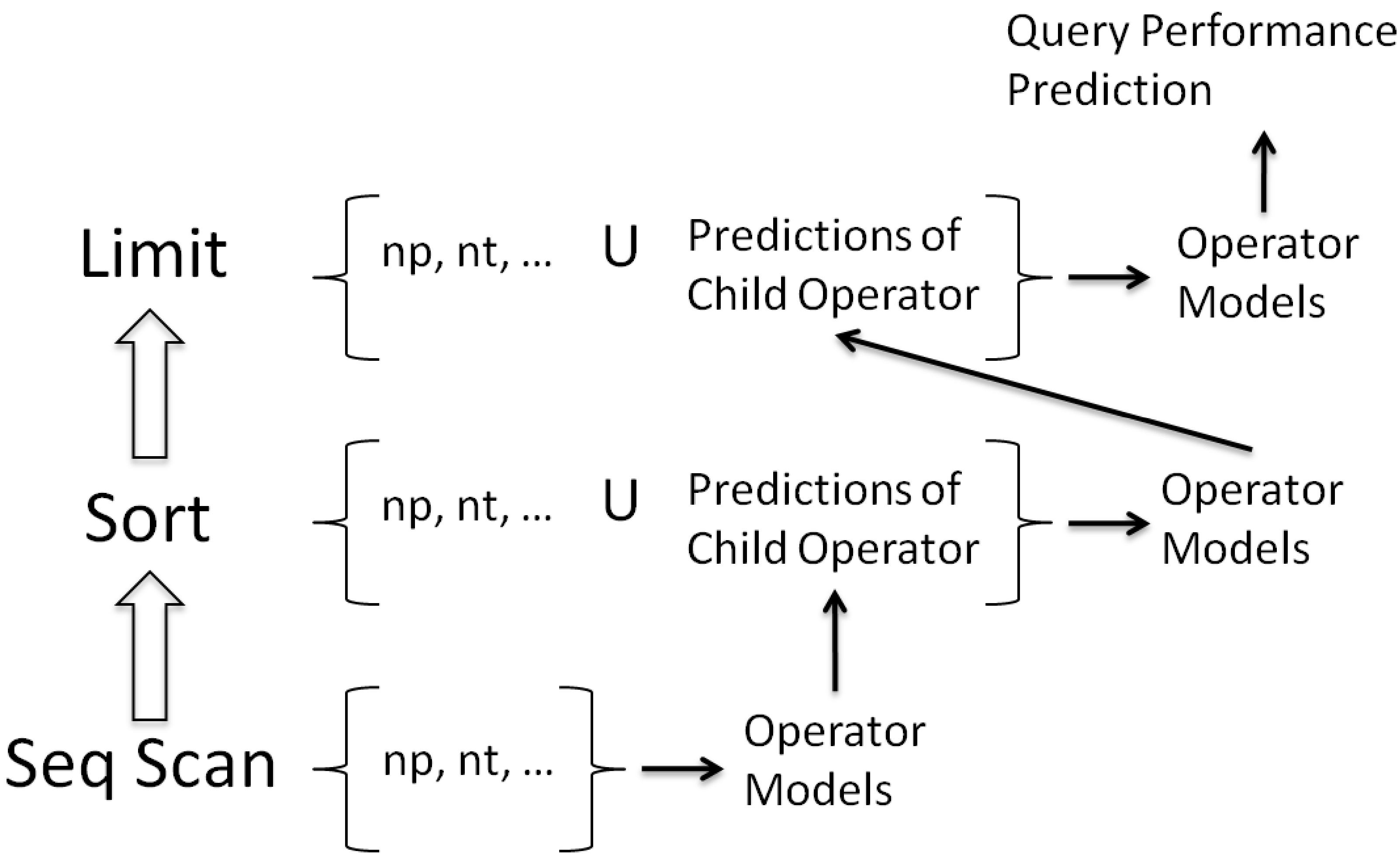
$$\text{subject to } \mathbf{E}[u^2] = 1$$

$$\mathbf{E}[v^2] = 1$$

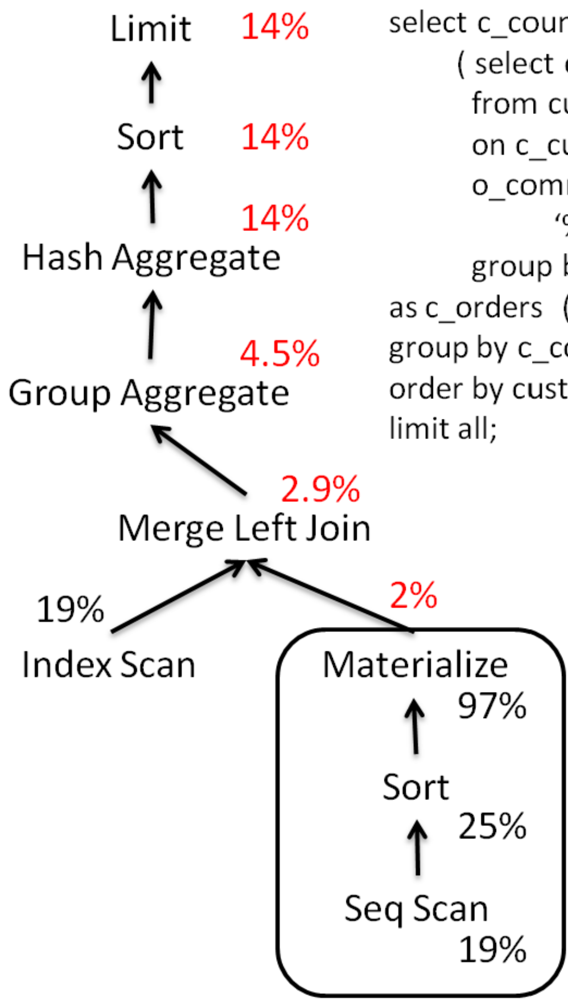
Related work



Related work

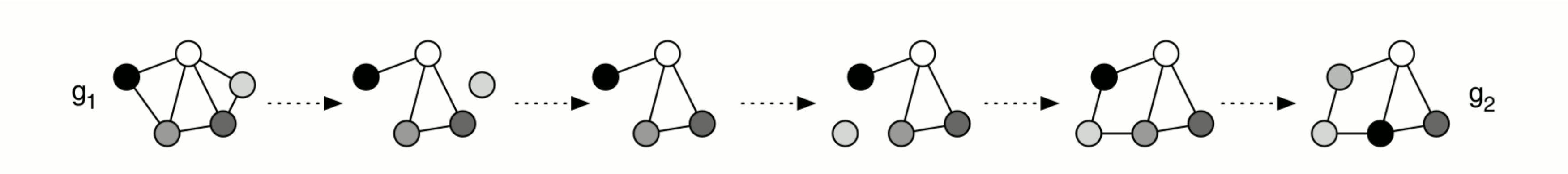


Related work




```
select c_count, count(*) as custdist from
        ( select c_custkey, count(o_orderkey)
          from customer left outer join orders
          on c_custkey = o_custkey and
          o_comment not like
                  '%pending%accounts%'
          group by c_custkey )
as c_orders  (c_custkey, c_count)
group by c_count
order by custdist desc, c_count desc
limit all;
```

Related work



...

And many other ways of feature construction

Related work

That was very interesting, but

$$\left\{ \underset{x \in all\_plans(query)}{\operatorname{argmin}} J(x) \middle| query \in Queries \right\}$$

Related work

- Towards predicting query execution time for concurrent and dynamic database workloads / W. Wu, Y. Chi, H. Hac´ıg¨um¨u¸s, J. F. Naughton
- Predicting query execution time: Are optimizer cost models really unusable? / W. Wu, Y. Chi, S. Zhu et al.
- Uncertainty aware query execution time prediction / W. Wu, X. Wu, H. Hacig¨um¨us, J. F. Naughton
- Sampling-based query re-optimization / Wu W., Naughton J. F., Singh H.

Related work

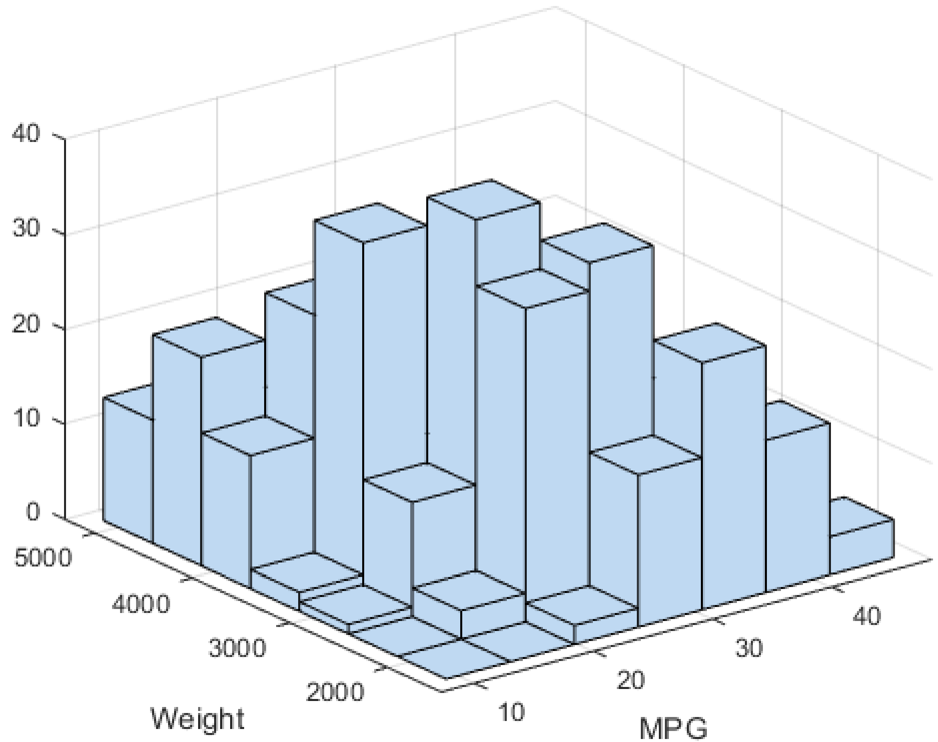
- Learning-based query performance modeling and prediction / M. Akdere, U. Cetintemel, M. Riondato et al.
- Predicting query execution time: Are optimizer cost models really unusable? / W. Wu, Y. Chi, S. Zhu et al.

But

- How good are query optimizers, really? / V. Leis, A. Gubichev, A. Mirchev et al.

Related work

Multidimensional histograms



Related work

- Selectivity estimation without the attribute value independence assumption. / Poosala V., 1997
- Selectivity estimation in extensible databases - a neural network approach / Lakshmi M. S., Zhou S., 1998
- Selectivity estimation using probabilistic models / Getoor L., Taskar B., Koller D., 2001
- A bayesian approach to estimating the selectivity of conjunctive predicates. / Heimel M., Markl V., Murthy K., 2009
- Cardinality estimation using neural networks / H. Liu, M. Xu, Z. Yu et al., 2015

K nearest neighbours

1. Define similarity between two objects:

$$\mathrm{dist}(\vec{x}_1, \vec{x}_2) = \ldots \qquad \mathrm{sim}(\vec{x}_1, \vec{x}_2) = \frac{1}{1 + \mathrm{dist}(\vec{x}_1, \vec{x}_2)}$$

2. Define K.

3. Find the K nearest objects and compute their weights:

$$w_i = \frac{\mathrm{sim}(\vec{x_{new}}, \vec{x_{(i)}})}{\mathrm{sim}(\vec{x_{new}}, \vec{x_{(1)}}) + \ldots + \mathrm{sim}(\vec{x_{new}}, \vec{x_{(K)}})}$$

4. Return weighted combination of their hidden variables:

$$y_{new} = w_1 y_{(1)} + \ldots + w_K y_{(K)}$$

Ridge regression

1. Model:

$$y_i \simeq w_1 \cdot x_{i,1} + \ldots + w_D \cdot x_{i,D} + b = f(\vec{x}_i, \vec{w}, b)$$

2. Fitting parameters:

$$L(\vec{w}, b) = \sum_{i=1}^{l} (f(\vec{x}_i, \vec{w}, b) - y_i)^2 + \lambda \sum_{i=1}^{D} w_i^2 \rightarrow \min_{\vec{w}, b}$$

3. Make predictions:

$$y_{new} \simeq f(\vec{x_{new}}, \vec{w}^{min}, b^{min}) = w_1^{min} \cdot x_{new,1} + \ldots + w_D^{min} \cdot x_{new,D} + b^{min}$$

Problem statement

Marginal selectivities:

1. 0.0001
2. 0.78
3. 0.23
4. 0.4
5. 0.5

l_partkey = p_partkey
AND
l_shipdate >= date '1995-12-01'
AND
l_shipdate < date '1995-12-01' + interval '1' month
AND
l_commitdate < l_receiptdate
AND
l_shipdate < l_commitdate

Joint selectivity

Information about data

List of conditions

Selectivity is 0.25!

Machine learning

Problem statement

Marginal selectivities:

1. 0.0001
2. 0.78
3. 0.23
4. 0.4
5. 0.5

l_partkey = p_partkey
AND
l_shipdate >= **const**
AND
l_shipdate < **const**
AND
l_commitdate < l_receiptdate
AND
l_shipdate < l_commitdate

Joint selectivity

Information about data

List of conditions

Selectivity is 0.25!

Machine learning

Problem statement

Selectivity	users.age > const	users.city = const	messages.sender_id = users.id
0.25	0.25	-	-
0.23	0.25	0.6	-
0.3	0.5	0.6	-
0.0005	-	0.5	0.001
...	...	...	...
???	0.5	0.5	-

Problem statement

LogSelectivity	users.age > const	users.city = const	messages.sender_id = users.id
-1.386	-1.386	0	0
-1.470	-1.386	-0.511	0
-1.204	-0.693	-0.511	0
-7.600	0	-0.693	-6.908
...	...	...	...
???	-0.693	-0.693	0

PostgreSQL model

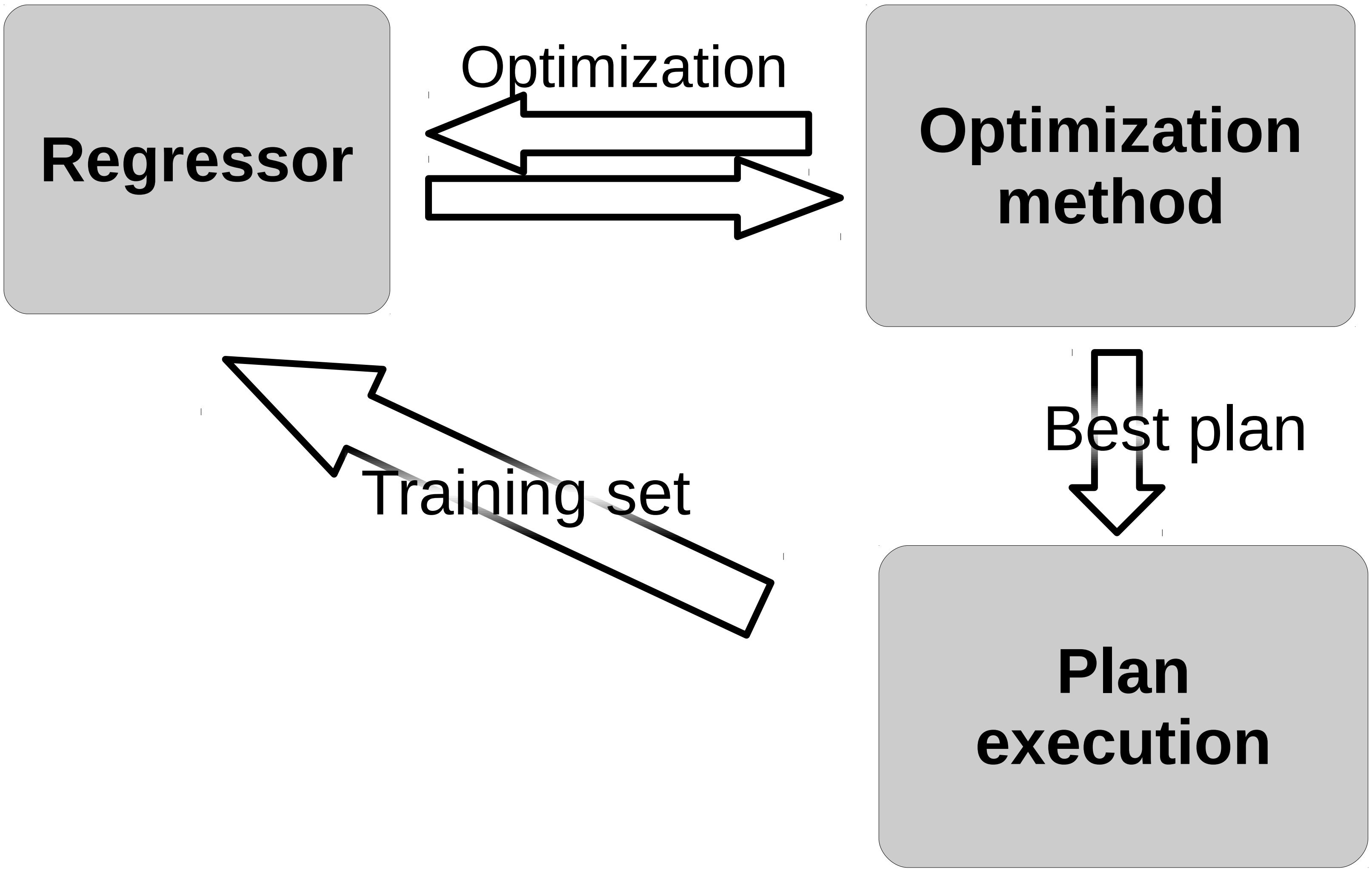
$$Joint\_selectivity = \prod_{c \in conditions} selectivity_c$$

$$\log Joint\_selectivity = \sum_{c \in conditions} \log selectivity_c$$

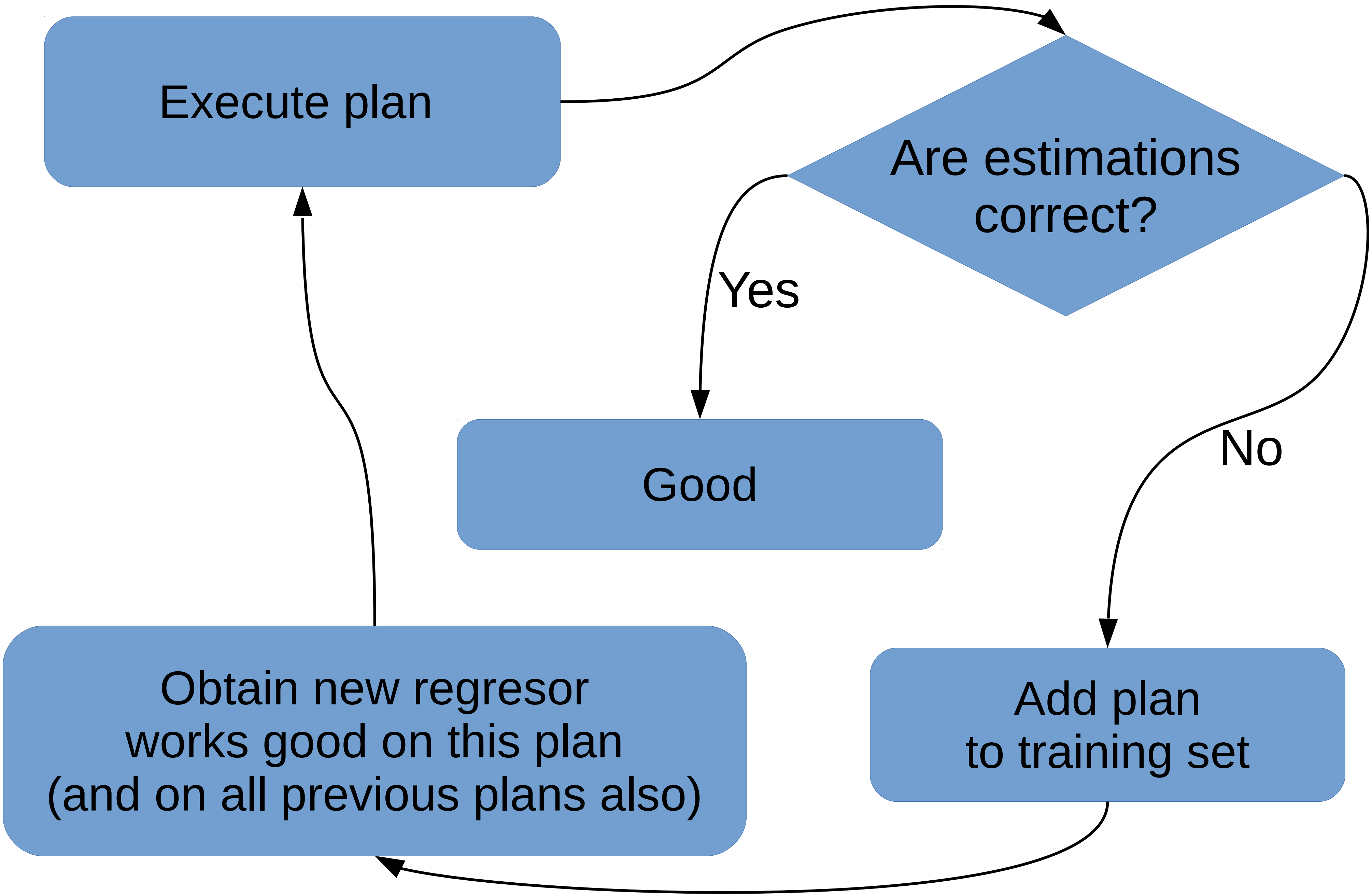
A special case of ridge regression:

$$\log Joint\_selectivity = \sum_{c \in conditions} w_c \log selectivity_c$$

Feedback


Regressor
Optimization
Optimization method
Best plan
Training set
Plan execution

Feedback


Execute plan
Are estimations correct?
Yes
No
Good
Add plan to training set
Obtain new regresor works good on this plan (and on all previous plans also)

Feedback

Does it converge?

What is convergence speed?

What guarantees on obtained plans or regressor do we have?

Feedback

Theorem 1

If regressor and its learning procedure fulfils follows:

- For each sample there is only one true selectivity
- Regressor predicts true selectivities for all samples from training set
- Duplicating a sample in training set doesn't change regressor

Then for a fixed number of queries and fixed data

- learning algorithm will converge (regressor and best plans are not changing) in finite number of steps
- predictions are correct for all conditions sets from executed plans

Feedback

Theorem 1

Does it converge?

Yes, in finite number of steps

What is convergence speed?

Don't know

What guarantees on obtained plans or regressor do we have?

Predictions are correct for all executed plans

Theorem 2
with exploration by random noise

If regressor and its learning procedure fulfils follows:

- Theorem 1 conditions
- Random independent noise added to regressor's predictions
- Probability density for each sample and each selectivity $s \in [0, 1]$ is greater than some positive ε.
- Each possible plan has nonzero probability to be choosen.

Then for a fixed number of queries and fixed data

- learning algorithm will converge (regressor and best plans are not changing) in finite number of steps with probability 1
- predictions are correct for all conditions set from all possible plans

Feedback

Theorem 2

with exploration by random noise

Does it converge?

Yes, in finite number of steps with probability 1

What is convergence speed?

Don't know

What guarantees on obtained plans or regressor do we have?

Predictions are correct for all executed plans

Choosed plans are globally the best plans available

Feedback

The tried techniques

- Ridge regression
  - stochastic gradient descent
- Composition of ridge regressions
  - stochastic gradient descent
  - the exact solution of linear algebraic equation system by Gauss
- K Nearest Neighbours
  - K = 1

Obtained results: convergence

Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

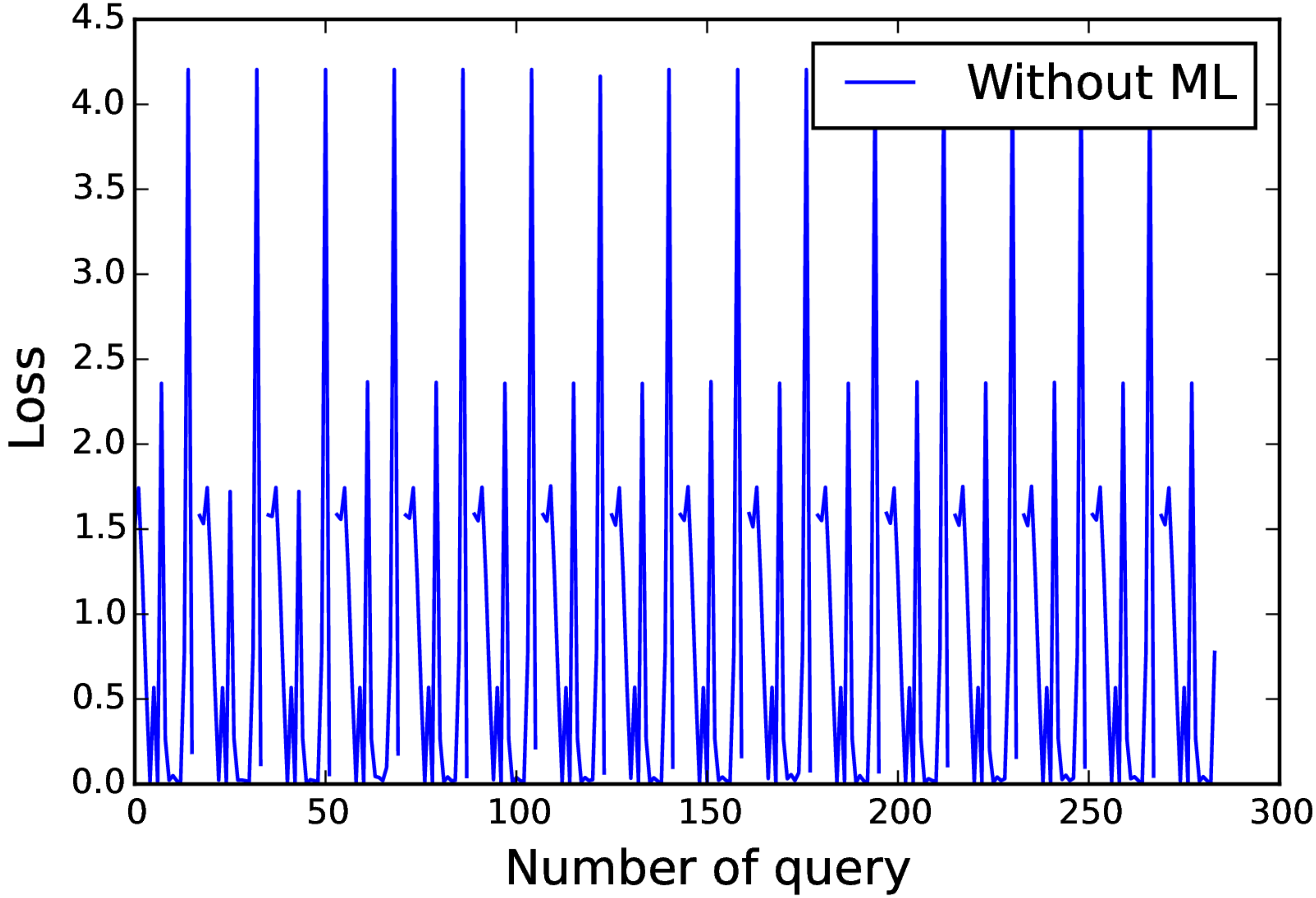
$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\log S_i - \log \hat{S}_i\right)^2}$$

N — number of nodes in plan
$\hat{S}_i$ — predicted selectivity of i-th node
S_i — true selectivity of i-th node

Obtained results: convergence

Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

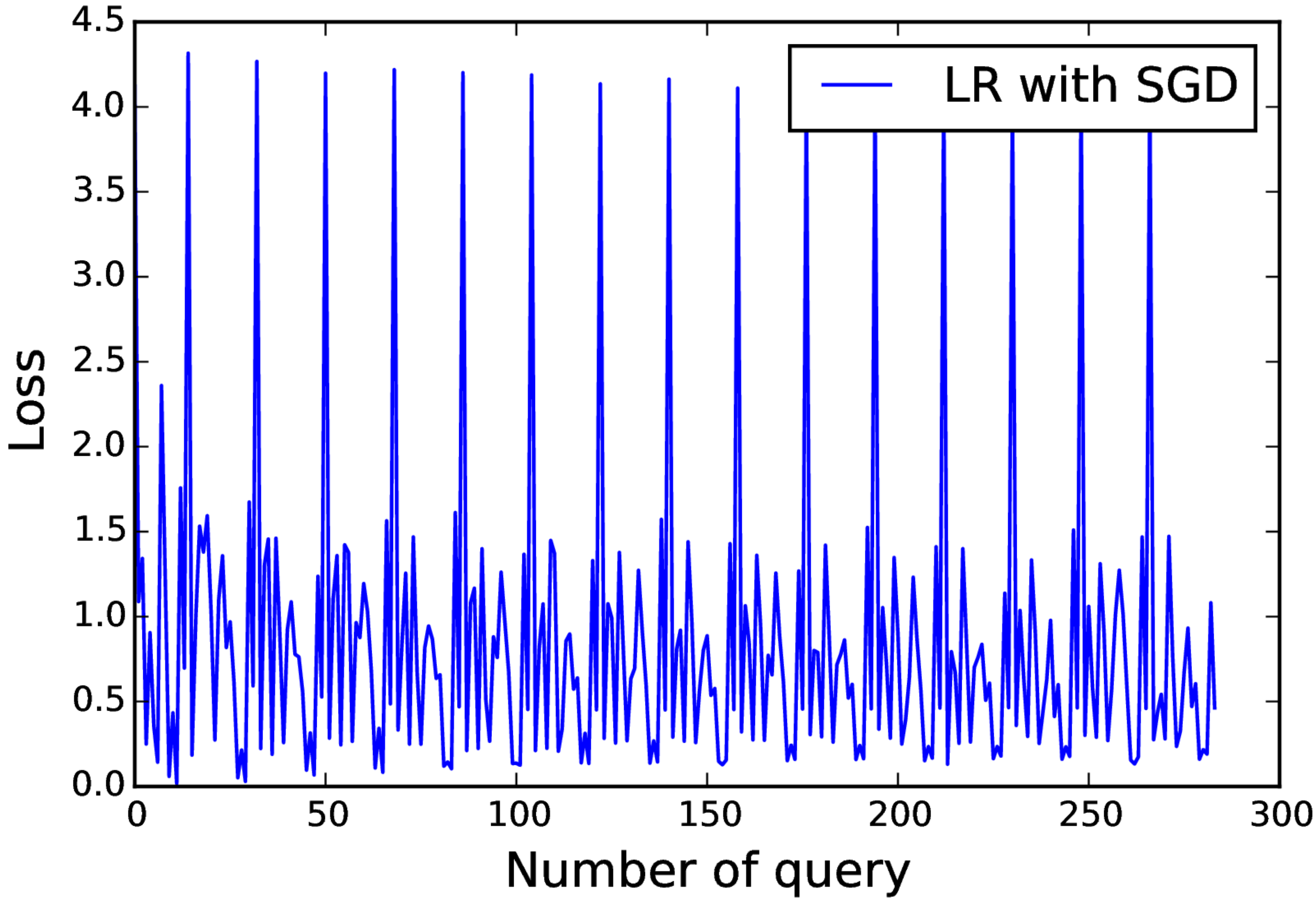
Mean quality: 0.87
Mean quality after 100 steps: 0.87



Obtained results: convergence

Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

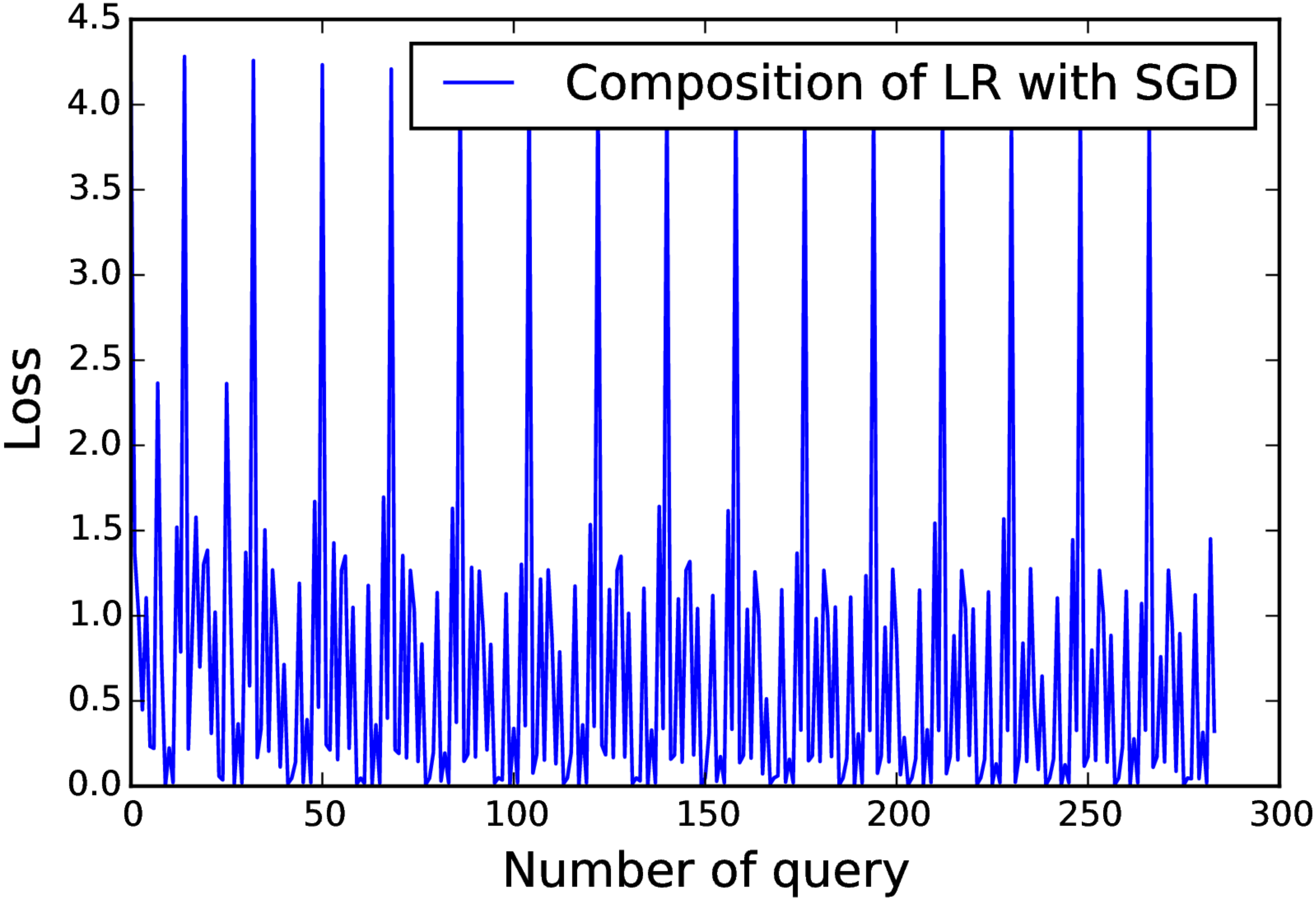
Mean quality: 0.87
Mean quality after 100 steps: 0.82



Obtained results: convergence

Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

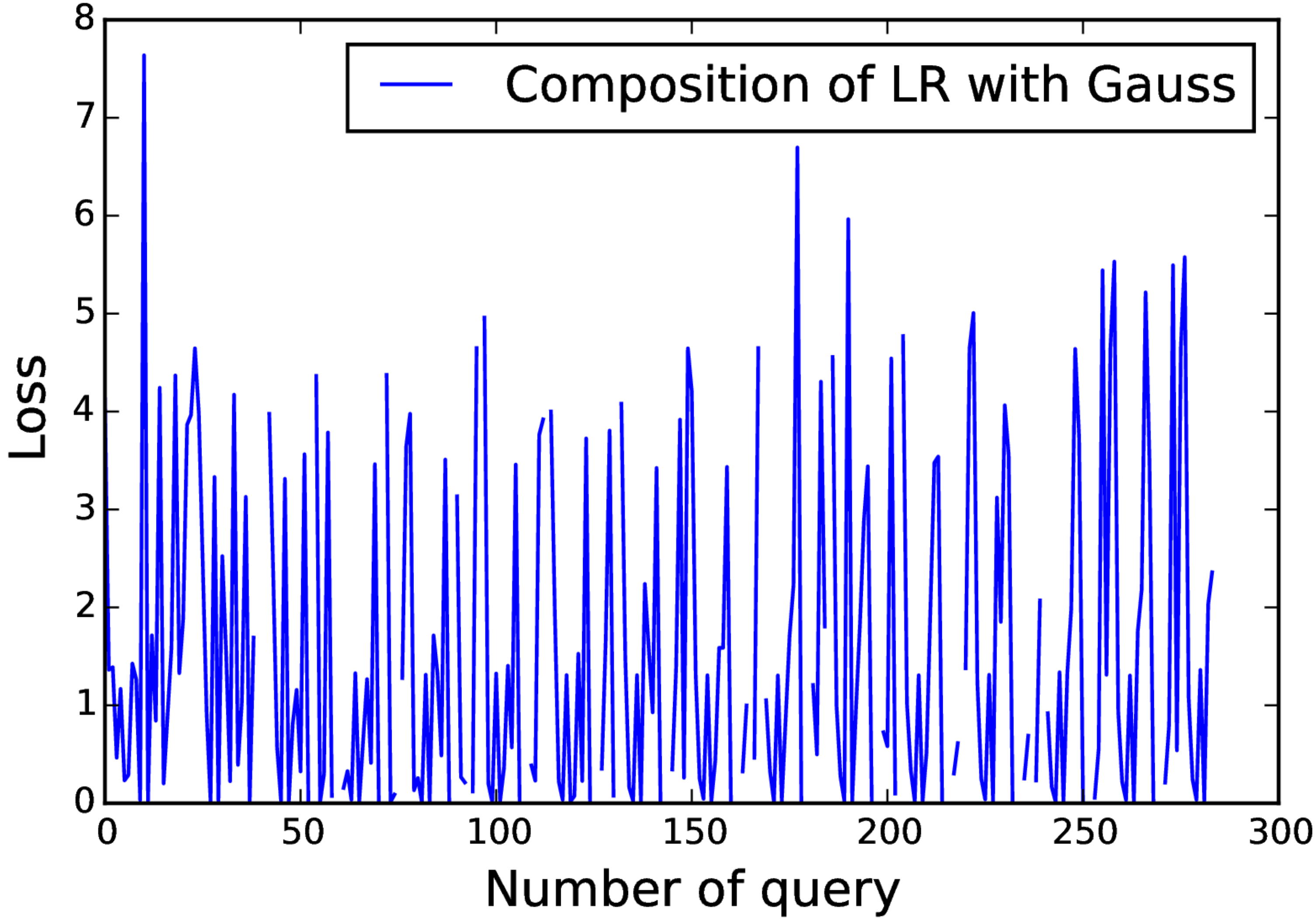
Mean quality: 0.72
Mean quality after 100 steps: 0.67



Obtained results: convergence

Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

Mean quality: 1.63
Mean quality after 100 steps: 1.61

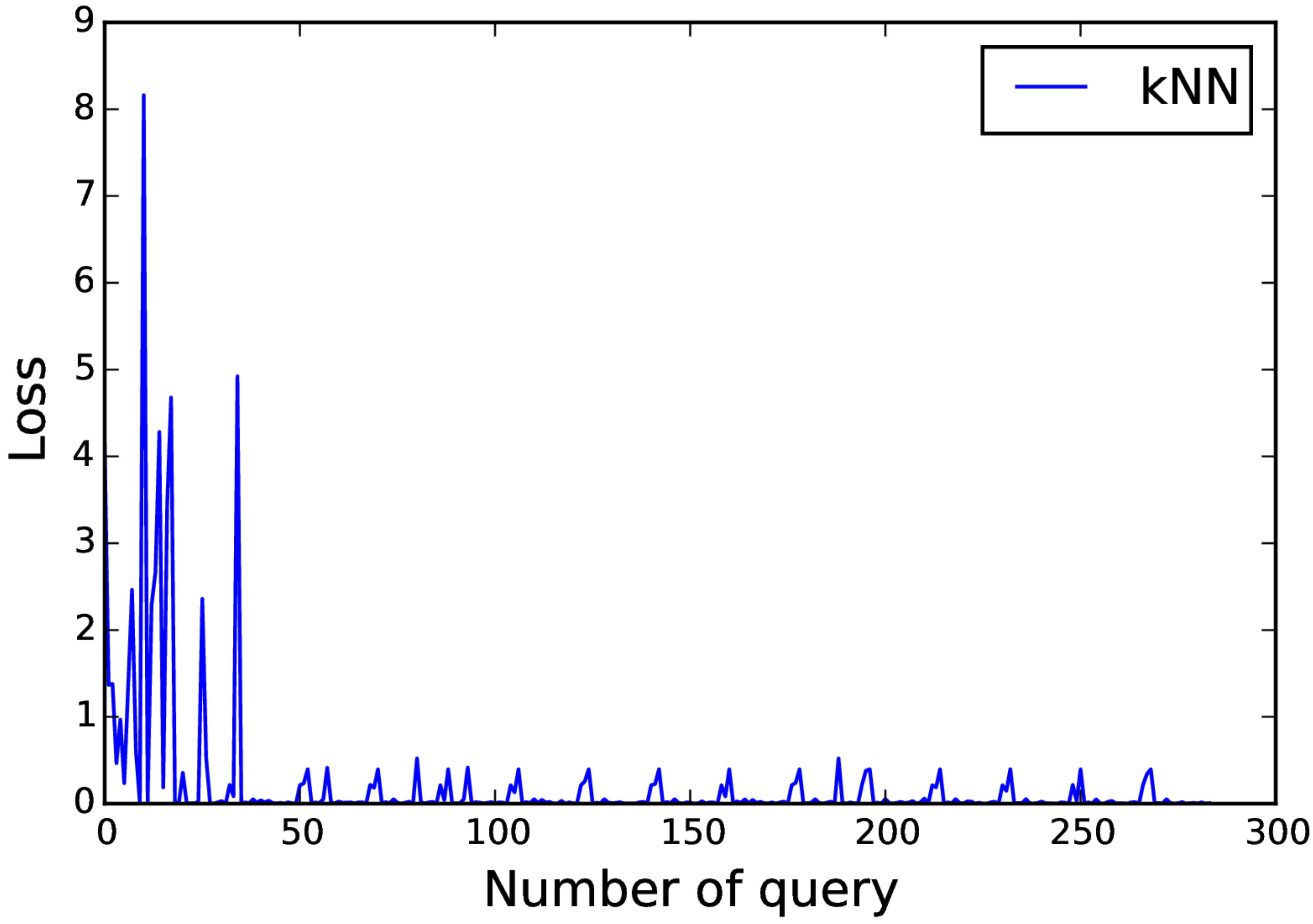


Obtained results: convergence

Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

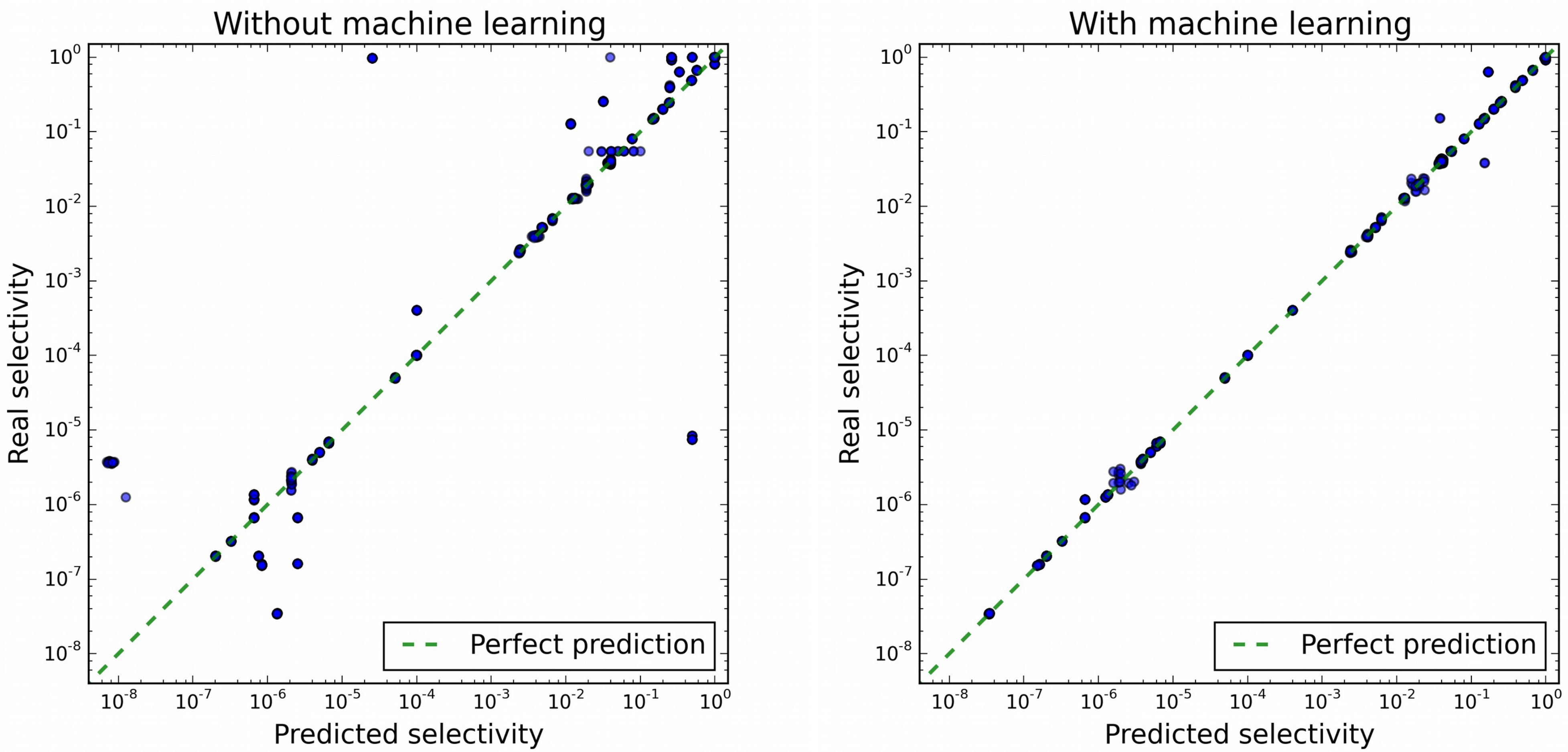
Mean quality: 0.23

Mean quality after 100 steps: 0.06

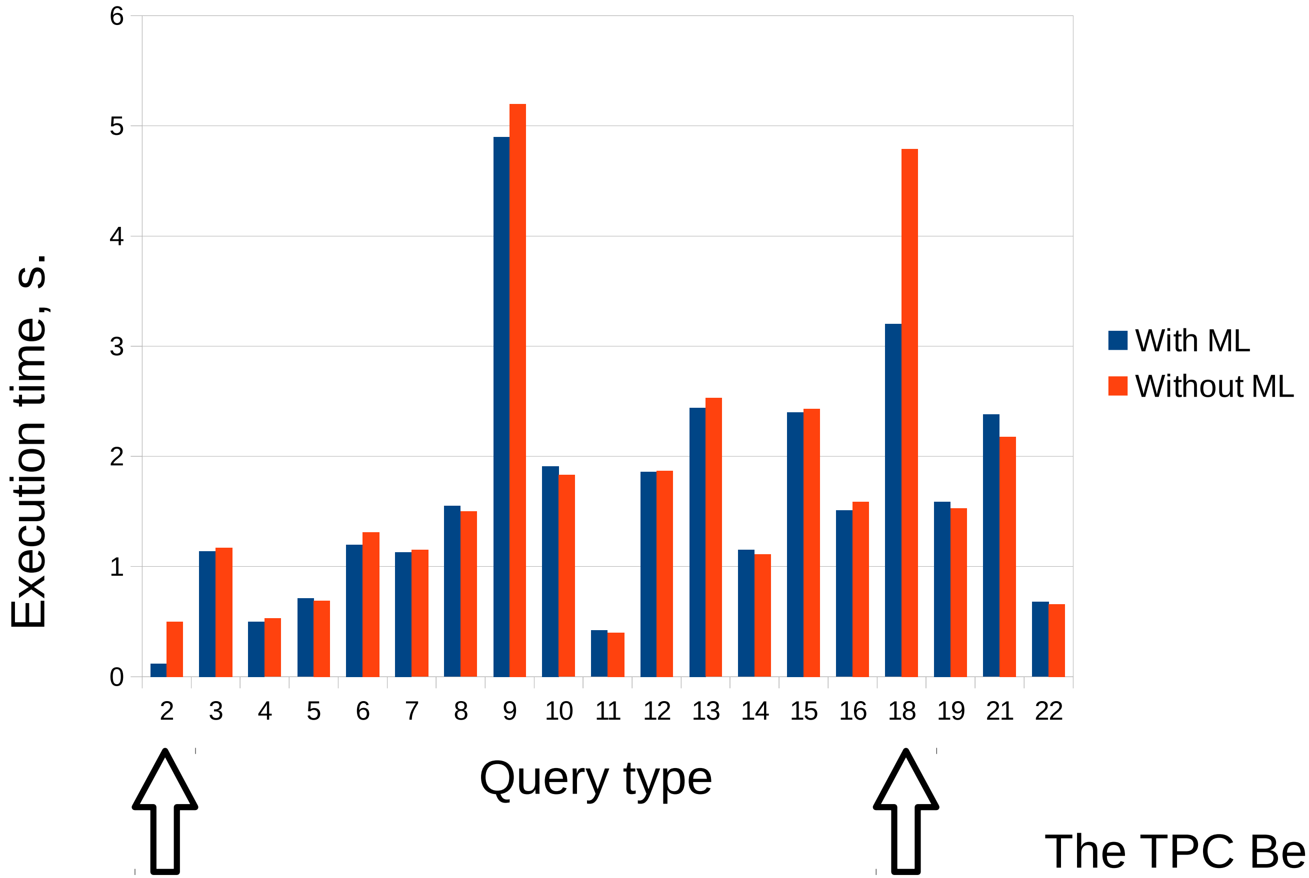


Obtained results: selectivity

Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/



Obtained results: performance



Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

Obtained results: issues

Marginal selectivities may be not precise enough

clauses.c | clausesel.c | costsize.c | execMain.c | instrument.c | instrument.h | execProcnode.c | pathnode.c | relnode.c | plancat.c | plannodes.h | copyfuncs.c | relation.h

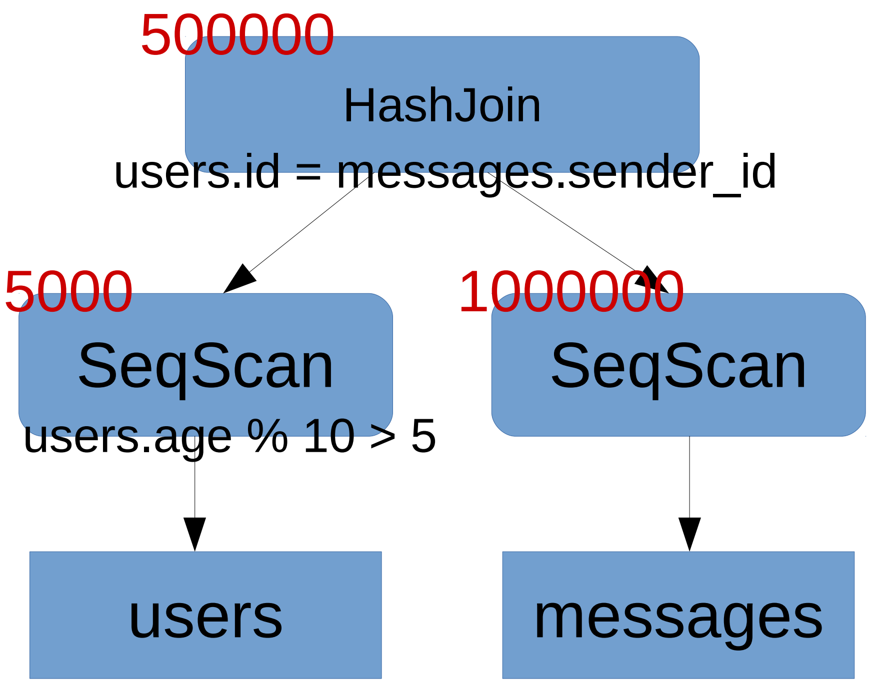
```c

        /*
         * DistinctExpr has the same representation as OpExpr, but the
         * contained operator is "=" not "<>", so we must negate the result.
         * This estimation method doesn't give the right behavior for nulls,
         * but it's better than doing nothing.
         */
        if (IsA(clause, DistinctExpr))
            s1 = 1.0 - s1;
    }
    else if (is_funcclause(clause))
    {
        /*
         * This is not an operator, so we guess at the selectivity. THIS IS A
         * HACK TO GET V4 OUT THE DOOR.  FUNCS SHOULD BE ABLE TO HAVE
         * SELECTIVITIES THEMSELVES.       -- JMH 7/9/92
         */
        s1 = (Selectivity) 0.3333333;
    }
#ifdef NOT_USED
    else if (IsA(clause, SubPlan) ||
             IsA(clause, AlternativeSubPlan))
    {
        /*
         * Just for the moment! FIX ME! - vadim 02/04/98
         */
        s1 = (Selectivity) 0.5;
    }
#endif
    else if (IsA(clause, ScalarArrayOpExpr))
    {
        /* Use node specific selectivity calculation function */
        s1 = scalararraysel(root,
                            (ScalarArrayOpExpr *) clause,
                            treat_as_join_clause(clause, rinfo,
                                                 varRelid, sjinfo),
                            varRelid,
```

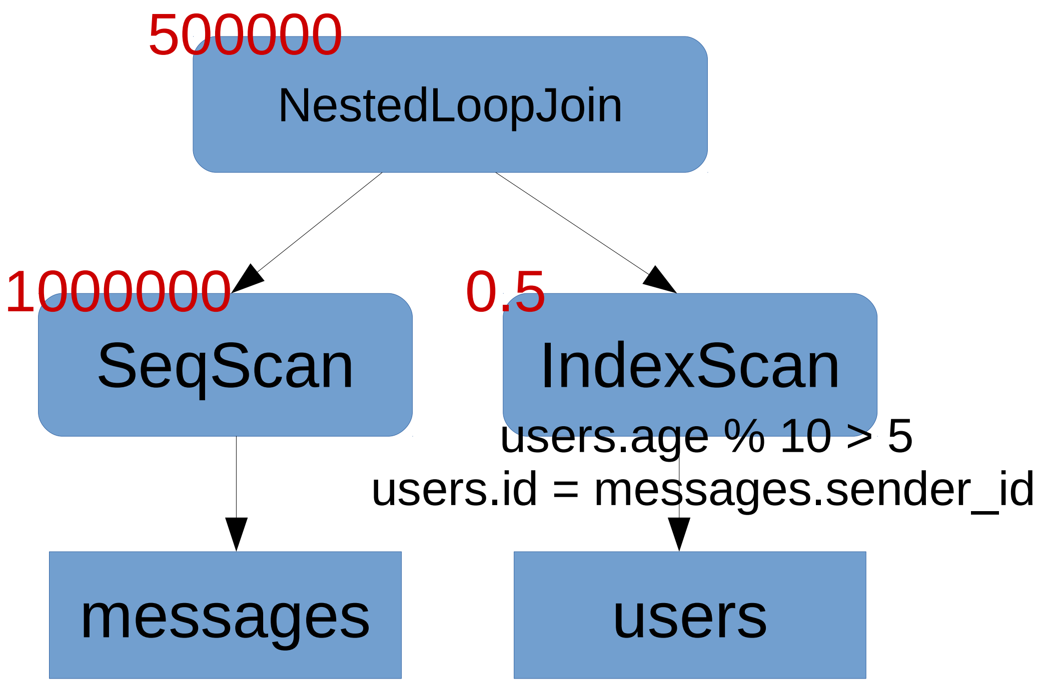
Obtained results: issues

```
SELECT *
FROM users, messages
WHERE users.id = messages.sender_id
AND users.age % 10 > 5;
```

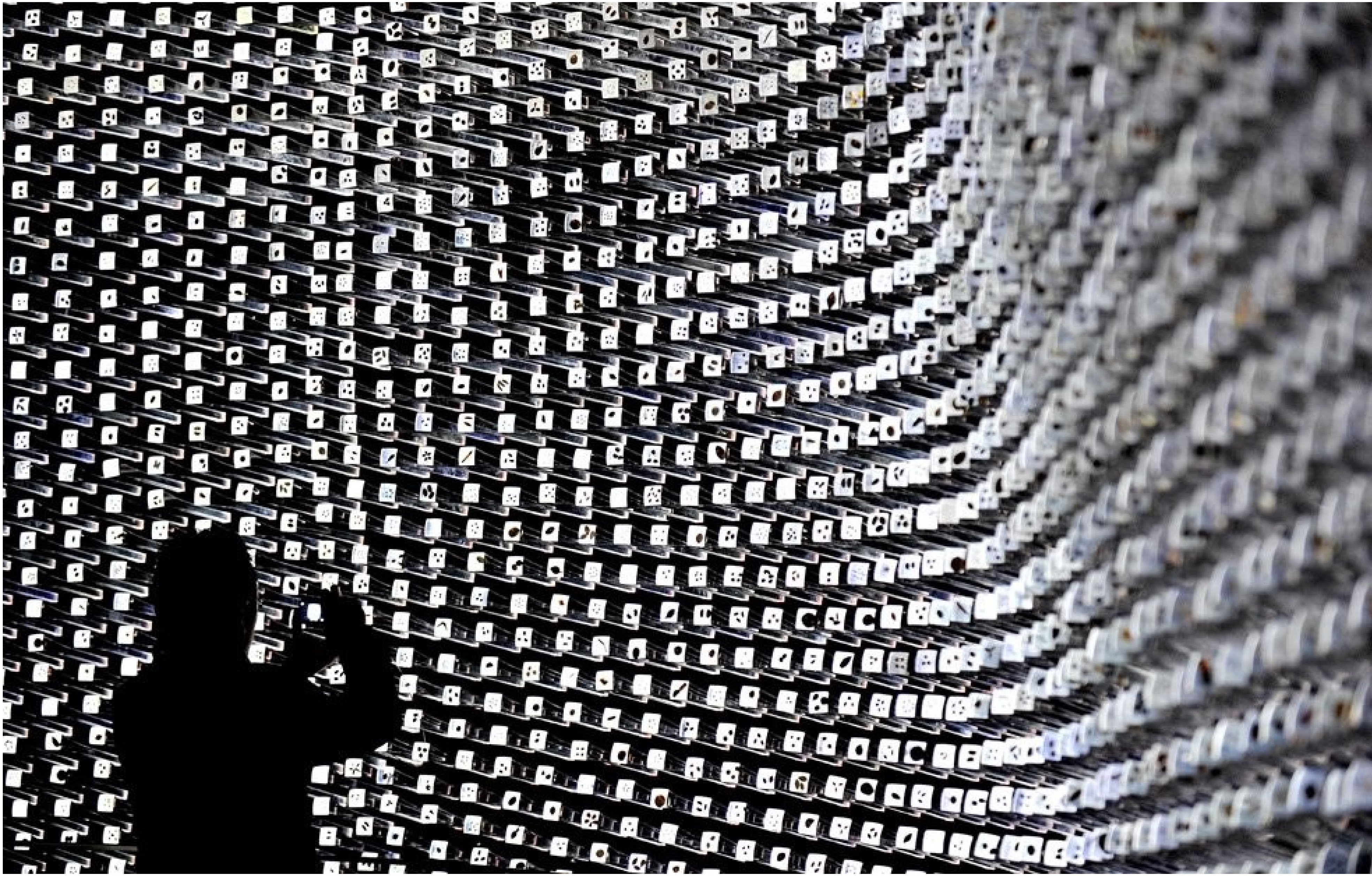
We have to predict:



We see:

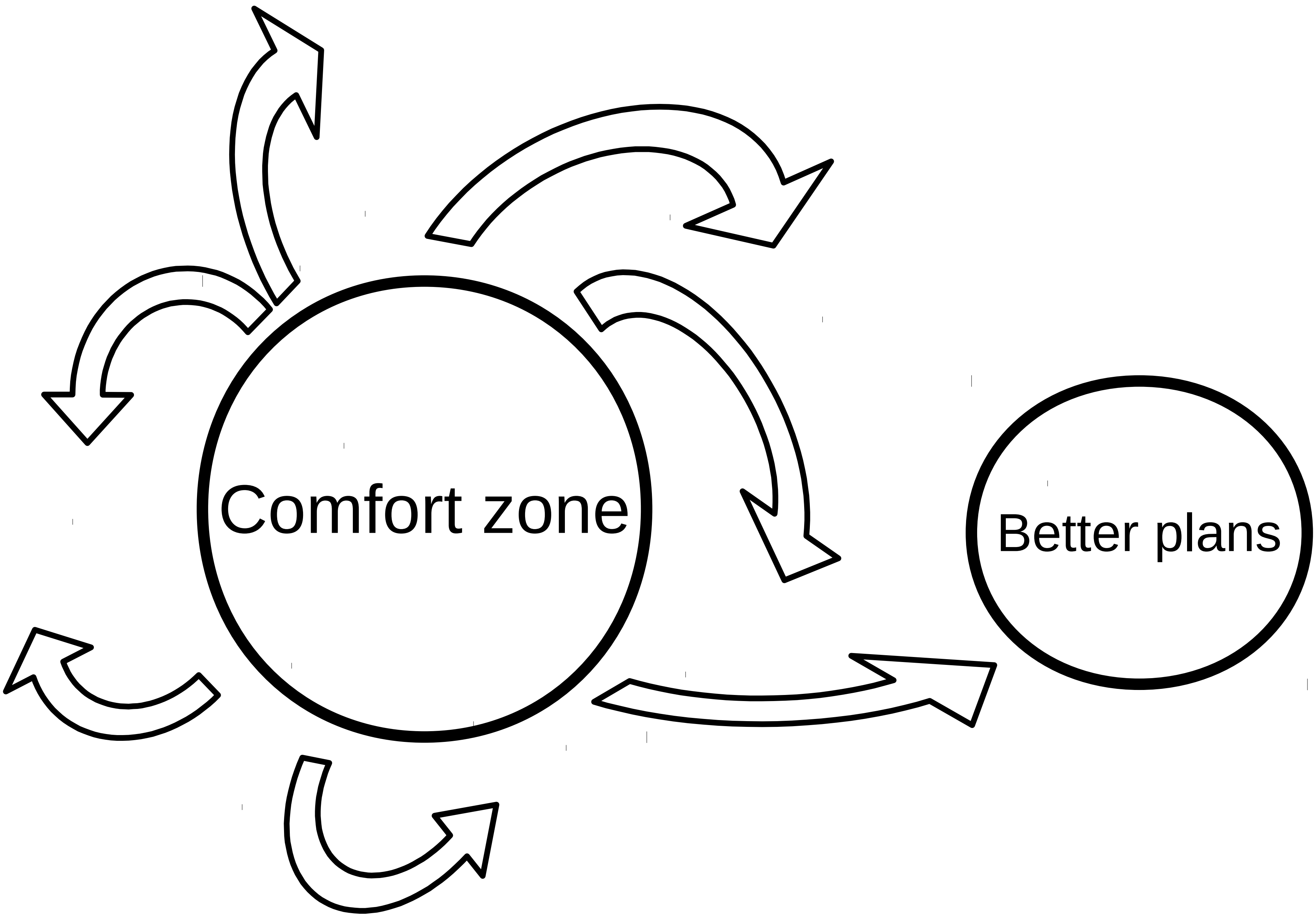


Sample selection



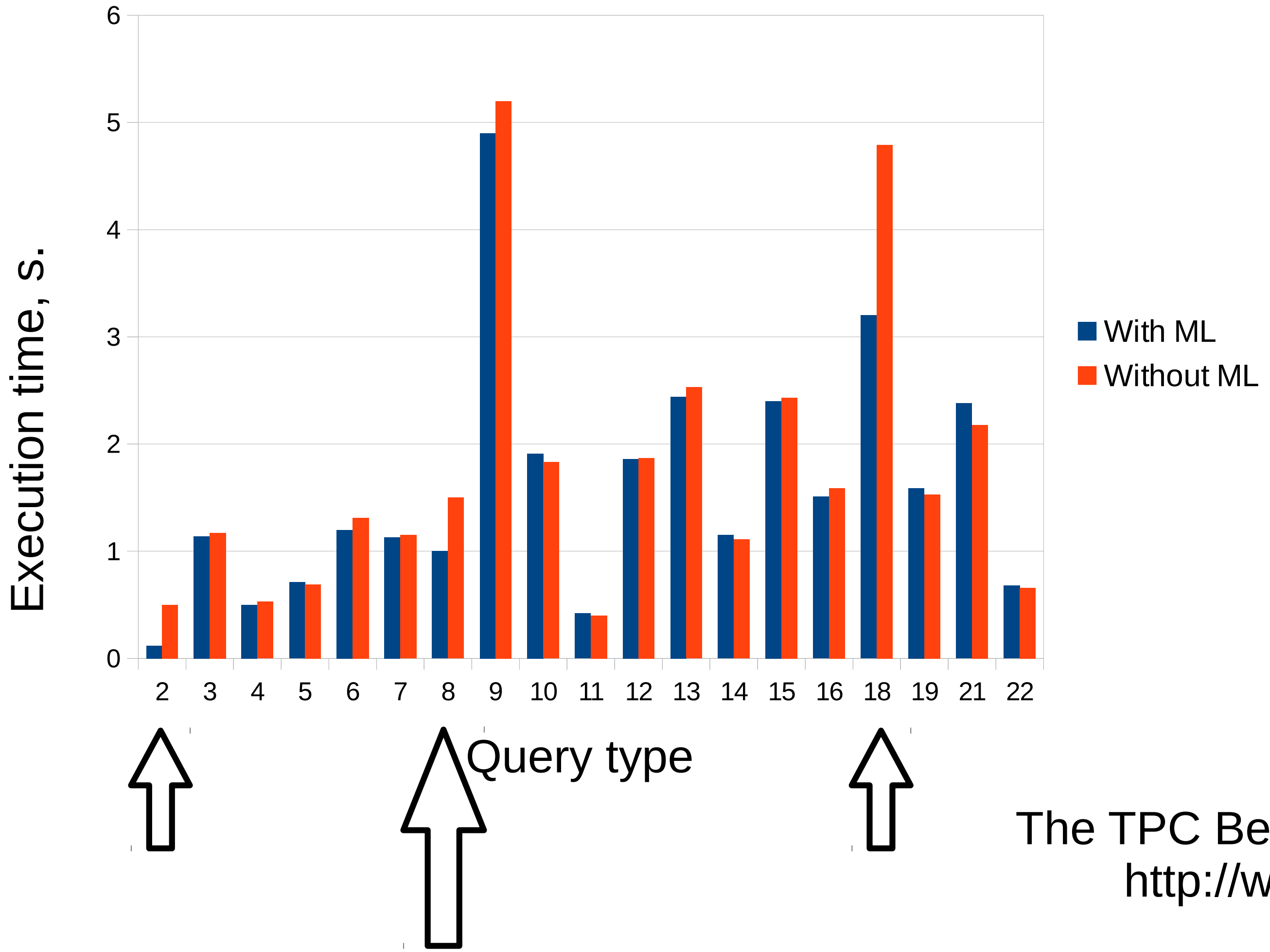
Realtime adaptation

Space of plans exploration



Space of plans exploration

Obtained results: performance acceleration



Dataset:
The TPC Benchmark™H (TPC-H)
http://www.tpc.org/tpch/

Questions?